



v3.0b0

Maximum likelihood large phylogeny estimation  
using the metapopulation genetic algorithm (MetaGA)  
& other stochastic heuristics

Manual version 3.1 (Feb 19, 2013)

Đorđe Grbić & Michel C. Milinkovitch  
*Lab. of Artificial & Natural Evolution (LANE),  
Dept of Genetics & Evolution,  
University of Geneva,  
Switzerland*  
[www.lanevol.org](http://www.lanevol.org)

Raphaël Helaers  
*Lab. of Human Molecular Genetics (GEHU)  
de Duve Institute, UCLouvain,  
B-1200 Brussels  
Belgium*

# TABLE OF CONTENTS

1. In a nutshell	3
2. Background	4
3. The metaGA algorithm & MetaPIGA	5
4. The software MetaPIGA	6
4.1. Availability	6
4.2. Recommended citations	6
4.3. CPU, GPU, Operating Systems, and memory requirements	6
5. Using MetaPIGA	10
5.1. Summary	10
5.2. Launching MetaPIGA & opening a file	11
5.3. [D] Dataset Settings	13
5.3.1 Overview	13
5.3.2 The 'Dataset' tab	14
5.3.3 The 'Codons' tab	16
5.4. [A] Analysis Settings	18
5.4.1. The 'Heuristic' tab	18
5.4.2. The 'Evaluation criterion' tab	21
5.4.3. The 'Starting tree(s)' tab	24
5.4.4. The 'Operators' tab	25
5.4.5. The 'Miscellaneous' tab	27
5.4.6. Exiting the Settings Window	30
5.5. [R] The Run window	31
5.6. [T] The tree viewer	33
5.6.1. Viewing and evaluating trees	33
5.6.2. Ancestral states reconstruction	34
5.7. Building and running batch files with the GUI	35
5.7.1. Transferring analysis settings among datasets	35
5.7.2. Duplicating datasets for batch files	35
5.8. Building batch files manually	36
5.9. The 'Tools' Menu	38
5.10. Troubleshooting	39
6. Acknowledgements	40
7. Appendix 1: The MetaPIGA commands	41
8. Appendix 2: Using the Stochastic Simulated Annealing (SSA)	53
9. Appendix 3: A simple introduction to ML phylogeny inference	56
9.1. Introduction	56
9.2. The General-Time-Reversible (GTR) Model	56
9.3. Computing the likelihood of a tree	57
10. Bibliography	60

# 1. In a nutshell

The development of heuristics implemented in robust application softwares has made large phylogeny inference a key step in most comparative studies involving molecular sequences. The choice of a phylogeny inference software is not only dictated by the raw performance (speed) of the algorithm(s) and of its (their) implementation, the availability of complex substitution models, and the accuracy of the resulting trees, but also by a combination of parameters pertaining to the ease-of-use and the availability of specific functionalities.

Here, we present the manual of MetaPIGA, a robust implementation of several stochastic heuristics for large phylogeny inference (under maximum likelihood), including a *Random-Restart Hill Climbing*, a *Stochastic Simulated Annealing* (SSA) algorithm, a classical *Genetic Algorithm* (GA), and the *Metapopulation Genetic Algorithm* (metaGA) together with complex substitution models, discrete Gamma rate heterogeneity, and the possibility to partition data. MetaPIGA handles nucleic-acid and protein datasets as well as morphological (presence/absence) data. The benefits of the metaGA ([1] [Lemmon & Milinkovitch 2002; PNAS, 99: 10516-10521](#)) are as follows: (i) it resolves the major problem inherent to classical Genetic Algorithms (*i.e.*, the need to choose between strong selection, hence, speed, and weak selection, hence, accuracy) by maintaining high inter-population variation even under strong intra-population selection, and (ii) it generates branch support values that approximate posterior probabilities.

The software MetaPIGA also implements:

- ✓ Simple dataset quality control (testing for identical sequences and excessively ambiguous or excessively divergent sequences);
- ✓ Automated trimming of poorly aligned regions using the *trimAl* algorithm [2];
- ✓ The Likelihood Ratio Test, Akaike Information Criterion, and Bayesian Information Criterion for the easy selection of nucleotide and amino-acid substitution models that best fit the data;
- ✓ Ancestral-state reconstruction of all nodes in the tree;
- ✓ Codon models for the analysis of protein-coding nucleotide sequences;
- ✓ Faster Likelihood computation on Nvidia graphics cards;
- ✓ Automated stopping rules based on convergence statistics.

MetaPIGA provides high customization of heuristics' and models' parameters, manual batch file and command line processing. However, it also offers an extensive and ergonomic graphical user interface and functionalities assisting the user for dataset quality testing, parameters setting, generating and running batch files, following run progress, and manipulating result trees.

MetaPIGA uses standard formats for data sets and trees, is platform independent, runs in 32- and 64-bits systems, and **takes advantage of multiprocessor and/or multicore computers**. Note that MetaPIGA allows the use of the [XtremWeb-CH](#) infrastructure for distribution of multiple jobs on a **Grid**.

MetaPIGA is freely available to academics at [www.metapiga.org](http://www.metapiga.org) and [www.lanevol.org](http://www.lanevol.org)

## 2. Background

Phylogeny inference allows, among others, detecting orthology/paralogy relationships among gene-family members (*e.g.*, [3-6]), estimating divergence times and evolutionary rates (*e.g.*, [7-9]), reconstructing ancestral sequences (*e.g.*, [10-14]), identifying molecular characters constrained by purifying selection or which experienced positive selection (*e.g.*, [15]), uncovering hidden biodiversity (*e.g.*, [16]), and mapping the evolution of morphological, physiological, epidemiological, biogeographical, and even behavioral characters [17, 18]. Molecular phylogeny inference is now a mature science, and an important part of the maturation process pertained to the realization (since the late 1990's) that the quest for the Holy Grail of *THE* absolute best tree should be abandoned for a much more meaningful goal: the inference of clades and trees robustness. Still, this objective remained intractable in practice because of (*a*) the *NP*-hard nature of optimality-criterion-based phylogeny inference (*i.e.*, no algorithm can solve it in polynomial time; [19, 20]) and (*b*) the large computing-time requirements when using complex substitution models (and rate heterogeneity across sites) in the framework of what has been identified as the probable most robust optimality criterion: Maximum Likelihood (ML; [21-23]; See Appendix 3 for an introduction to ML). Today large phylogeny inference is incorporated, across biological disciplines, as an essential step in most comparative studies involving nucleotide or protein sequences. This has been made possible thanks to both theoretical and practical developments.

First, one key advance that made large phylogeny inference tractable is the implementation in this field of stochastic heuristics with inter-step optimization, *i.e.*, a family of approaches that existed for decades in physics and computer science and explore multidimensional solution spaces in a much more efficient manner than the older intra-step optimization hill-climbing methods. Indeed, in the latter, one prime parameter (typically, the topology of the tree) is modified and all other parameters are optimized before the new solution is evaluated whereas, in stochastic heuristics, all free parameters are optimized while the search proceeds. Inter-step optimization methods include Markov Chain Monte Carlo (MCMC) approximations of the Bayesian approach [24, 25], stochastic simulated annealing [26], and genetic algorithms [1, 27-30]. The efficiency of stochastic heuristics is quite counterintuitive but can be explained by several factors: (*a*) poorer solutions are accepted with a non-null probability (contrary to hill-climbing that strictly restricts moves toward better likelihood values) such that valleys in likelihood space can eventually be crossed; and (*b*), parameters are not over-optimized (*e.g.*, starting and intermediate trees are generally largely sub-optimal, hence, optimizing model parameters on these trees is a clear example of over-fitting). In addition, we think that avoiding over-optimization at every topology evaluation generates a flatter likelihood-space shape, such that valleys are more easily crossed and local optima more easily escaped. This suggestion however requires further investigation.

Second, several stochastic methods have been incorporated into robust application softwares. The importance of that point should not be underestimated. For example, the success of Bayesian methods is probably due as much to its incorporation into robust and efficient software (*e.g.*, MrBayes; [31]) as to the theoretical appeal of generating marginal posterior probabilities [25]. The software RaxML [32], enjoys deserved popularity because it is one of the fastest ML phylogeny inference programs available to date (despite that it does not incorporate stochastic methods) thanks to the implementation of approximations to rate heterogeneity across sites and smart computer science tricks speeding up likelihood computation: optimized parallel code and 'Subtree Equality Vectors' (*i.e.*, the extension of character compression to the subtree level). Similarly, highly efficient parallel code has recently been implemented for the evaluation of phylogenies on graphics processing units (GPUs), resulting in 10 to 100-fold speed increase over an optimized CPU-based computation [33]. This efficient use of new hardware, existing stochastic heuristics (in this case, an MCMC approach in a Bayesian framework), and smart code parallelization for efficient harnessing of the hundreds of GPU processing cores allowed the authors to use a 60-state codon model on a

dataset of 62 complete mitochondrial genomes. Note that MetaPIGA now implements GPU computation (since version 3.0b0).

The availability of multiple excellent softwares implementing different robust heuristics is clearly an asset for the end user: reliable results might be identified because they remain stable across softwares and methods. However, many users chose one single main software for their analyses, and this choice is sometimes dictated by availability of functionalities of importance (*e.g.*, batch analyses, GTR nucleotide substitution model [34] and rate heterogeneity [35-37], possibility to partition data) but that do not pertain to the performances of the specific heuristic implemented. Finally, given that the need to infer large trees is critical in multiple biological disciplines, the non-specialist can be baffled by the large number of available heuristics, parameters, and softwares, such that the most user-friendly tools are sometimes preferred even if more robust or more efficient (but less user-friendly) softwares are available.

There is therefore a challenge to supply softwares that are both easy to use for the non-specialist, provide flexibility for the specialist, and allow fast and robust inference for both. We hope MetaPIGA version 3 provides a solution to this conundrum.

### 3. The metaGA algorithm & MetaPIGA

The Metapopulation Genetic Algorithm (MetaGA; [1]) is an evolutionary computation heuristic in which several populations of trees exchange topological information which is used to guide the Genetic Algorithm (GA) operators for much faster convergence. Despite the fact that the metaGA had initially been implemented in a simple and unoptimized software (metaPIGA-v1) together with simple nucleotide substitution models, an approximate rate heterogeneity method, and only a low number of functionalities, it has been suggested as one of the most efficient heuristics under the ML criterion. Furthermore, multiple metaGA searches provide an estimate of the posterior probability distribution of trees [1].

**The metaGA resolves the major question inherent to classical GA approaches: should one use a soft or a stringent selection scheme? Indeed, strong selection produces good solutions in a short computing time but tend to generate sub-optimal solutions around local optima. Conversely, mild selection schemes considerably improve the probability to escape local optima and find better solutions, but greatly increase computing time. As the metaGA involves several parallel searches, initial inter-population variation can be very high (especially if random or pseudo-random starting trees are used), and somewhat maintained during the search, even under extreme intra-population selection.**

Although the metaGA has been shown to perform very well [1, 38, 39] it initially did not implement complex substitution models, discrete Gamma rate heterogeneity, and the possibility to partition data. Here, we present MetaPIGA version 3, a program in which we performed such an implementation, both for nucleotide and protein data, together with a hill climbing, a classical Genetic Algorithm (GA), and a Stochastic Simulated Annealing (SSA) algorithm. MetaPIGA version 3 also implements dataset quality control, automated trimming of poorly aligned regions, criteria (Likelihood Ratio Test, Akaike Information Criterion, and Bayesian Information Criterion) for the easy selection of nucleotide and amino-acid substitution models that best fit the data, ancestral-state reconstruction of nodes, Codon models for the analysis of protein-coding nucleotide sequences, faster Likelihood computation on Nvidia graphics cards, and automated stopping rules based on convergence statistics. MetaPIGA can also be parallelized on a Grid of computers.

MetaPIGA gives access both to high parameterization, as well as to an ergonomic interface and functionalities assisting the user for sound inference of large phylogenetic trees.

## 4. The software MetaPIGA

### 4.1. Availability

The software MetaPIGA is freely available to academics at [www.metapiga.org](http://www.metapiga.org), and is available for Windows, Mac OSX, and Linux. Note that, each time you launch MetaPIGA, it checks for the availability of updates. MetaPIGA will always request your authorisation to perform such an update. This manual is also available in the MetaPIGA *help* menu.

**Disclaimer.** MetaPIGA is provided without warranty of any kind. The authors and their institutions do not warrant guarantee, or make any representation regarding the use or the results of the program or manual in terms of their correctness, reliability, or otherwise. In no case will the authors and their respective institutions be liable for any direct, special, indirect, incidental, consequential, or other damages arising from using the metaGA and/or any version of MetaPIGA and/or this manual and/or any supporting material. MetaPIGA is freely available only to Academics. If you are working for a commercial company and are planning to use MetaPIGA, please, contact [michel.milinkovitch-at-unige.ch](mailto:michel.milinkovitch-at-unige.ch)

### 4.2. Recommended citations

The Consensus Pruning (CP) and the Metapopulation Genetic Algorithm (metaGA) were originally described in the first reference below, whereas the version 2 of MetaPIGA (the software implementing the MetaGA and other heuristics) is described in the second. Hence, we would be grateful if you could cite these two references when publishing results produced with MetaPIGA version 3.

- ✓ Lemmon A.R. & M. C. Milinkovitch  
*The metapopulation genetic algorithm: an efficient solution for the problem of large phylogeny estimation*  
[Proceedings of the National Academy of Sciences \(PNAS\), USA, 99: 10516-10521 \(2002\)](#)
- ✓ Helaers R. & M. C. Milinkovitch  
*MetaPIGA v2.0: maximum likelihood large phylogeny estimation using the metapopulation genetic algorithm and other stochastic heuristics*  
[BMC Bioinformatics 2010, 11: 379](#)

### 4.3. CPU, GPU, Operating Systems, and memory requirements

**CPU & Operating Systems.** As optimality-criterion phylogeny inference in general, and ML inference in particular, is a computer intensive endeavour, fast CPUs are always preferable, even when using powerful heuristics such as MC<sup>3</sup> or the metaGA. Using a ranid frog dataset (provided with the software as one of the example datasets) of 64 taxa X 1976 nucleotides each, a typical metaGA run (4 populations of 4 individuals, and default parameter values) will take approximately 2 minutes to complete under a simple model (Jukes-Cantor) and about 20 minutes under a complex model (GTR + gamma distributed rate heterogeneity) on a single core of a 2.27 GHz Intel Xeon processor (you can easily reduce running time by distributing replicates on several cores, see below). Hence, when using datasets of over 100 taxa and when performing replicates (to estimate posterior probabilities of clades; see below), you should expect runs to last several hours. If you are experienced in the use of MrBayes [31], take as a rule of thumb that a thorough analysis using the MetaGA requires a running time similar to that of using MrBayes with the same dataset.

MetaPIGA is written in Java 1.6 such that the single code runs on 32 and 64-bits platforms under MacOS X, Linux, and Windows. We use the Java Multi-Threading technology to take advantage of multiprocessor and/or multicore computers, such that some tasks can be run in parallel. As replicates are independent, they are particularly prone to parallelization: different replicates can be assigned to any number of different processor cores (typically 4 - 12 in most 2013 machines). In addition, the metaGA heuristic itself is well suited to parallel implementation because many proc-



esses (mutations, selection, and likelihood computation) are independent across populations. Hence, different metaGA populations can be distributed to different processor cores. Parallelization of metaGA populations can be combined with parallelization of replicates (e.g., 16 cores allow running simultaneously 4 metaGA replicates with 4 populations treated simultaneously at each replicate). Note that distributing different replicates to different cores is more efficient (in terms of computation speed-up) than distributing different populations to different cores<sup>1</sup>. Hence, parallelization of populations usually increases running speed by about  $0.3n$  whereas parallelization of replicates increases running speed by almost  $n$  (where  $n$  = the number of CPU cores you assigned to MetaPIGA).

**Computing on GPU (Graphics cards).** Analyses of protein or codon datasets are particularly long because of the high number of possible state substitutions (20x20 for amino-acid data; 64x64 for Codon data). In such cases, performance can be substantially increased if likelihood computation is performed on GPUs (Graphics processing units), also called ‘Graphics cards’. These are devices that provide fine-grained parallelization. MetaPIGA version 3 can run on CUDA-capable graphics cards from Nvidia Corporation. The graphics card’s compute capability has to be at least 2.0. The list of CUDA-capable graphics cards can be found on the following web site: <https://developer.nvidia.com/cuda-gpus>. Note that the performances of GPUs are low for nucleotide sequence data, substantial for protein sequence data, and spectacular for codon sequence data.

In order to make use of the available supported graphics card, appropriate CUDA drivers have to be installed. The drivers and the installation instructions can be found on the following web site:

<https://developer.nvidia.com/cuda-toolkit-42-archive>. Be sure to install the 4.2 Toolkit version and the drivers that come with that version of the CUDA Toolkit. MetaPIGA v.3 hasn’t been tested on the newer versions of the CUDA Toolkit.

If you’re using a Linux distribution with graphics card, prior to launching MetaPIGA, you must set the environment variable that points to the CUDA library, like this:

```
export LD_PRELOAD={path to the CUDA library}:$LD_PRELOAD
```

Where {path to the CUDA library} points to the ‘libcuda.so’ CUDA library.

For example on one of our machines this variable setting looks like this:

```
export LD_PRELOAD=/usr/lib/nvidia-current/libcuda.so:$LD_PRELOAD
```

For best performances, the graphics card must have enough built-in memory (see the ‘**memory**’ sections below).

**The Grid.** If you are a user of the [XtremWeb-CH](#) infrastructure, you can use a Grid to perform your data analysis with coarse grained parallelization. This means that different replicates are computed on the different worker computers on the Grid. If you have 100 computers on your grid, your analysis will be about 100 times faster.

In order to use the grid, first you have to have an account on the XWCH. After you make an account, you have to ask the XtremWeb-CH support to connect a MetaPIGA module to your account. When the MetaPIGA module is ready, you have to upload the MetaPIGA binaries to your MetaPIGA module. Provided with MetaPIGA is a small program that uploads these binaries to the grid. This program is available in the MetaPIGA base folder on your computer in the subfolder ‘XWCH\_bin\_uploader’. You will have to provide the ‘MetaPIGA 3.jar’ that is in the base MetaPIGA folder, your user identification number, the grid server address, and the MetaPIGA module ID. These informations can be found in your XtremWeb-CH interface. If you can’t find them, consult with the XtremWeb-CH project people. Note that, every time MetaPIGA is updated, you will have to upload the binaries again in order to have the latest version of the MetaPIGA on the grid. For the user documentation, please, refer to the following web site:

[http://www.xtremwebch.net/mediawiki/index.php/How\\_use](http://www.xtremwebch.net/mediawiki/index.php/How_use)

**Memory.** Computing and storing the likelihood of large trees require large amounts of Random-Access Memory (RAM). Note that 32-bits systems can allocate a maximum of ~2Gb of memory to the Java Virtual Machine (JVM), whereas 64-bits systems are limited only by the amount of memory installed on the computer (the theoretical limit is 16 billions gigabytes). The

---

<sup>1</sup> Indeed, under CP, different populations of a single metaGA search must exchange topological information, hence, the running time at each generation is limited by the population which is slowest to complete. On the other hand, different replicates are totally independent.

equation below allows calculating the number of Giga-bytes of free RAM (*i.e.*, RAM that must be available when your OS is running) you will need for using MetaPIGA:

$$RAM(Gb) = \frac{T_r \cdot N \cdot D \cdot C \cdot S \cdot P_r \cdot 4}{1024^3}$$

where  $T_r$  is the number of trees used at each generation,  $N$  is the number of nodes in the tree ( $=2T-1$ , where  $T$  is the number of taxa),  $D$  is the number of data patterns<sup>2</sup>,  $C$  is the number of discrete categories of the gamma distribution (typically, 4), and  $S$  is the number of possible character states ( $S=4$ ,  $S=20$ , and  $S=64$  for DNA, protein, and Codon characters, respectively).  $P_r$  is the number of CPU cores assigned to the parallelization of replicates: doubling the number of CPU cores assigned to different replicates doubles the speed of the search but also doubles the amount of required RAM.

The number of trees ( $T_r$ ) used at each generation by MetaPIGA depends on the heuristic chosen:

- ✓  $T_r = 3$  for 'Hill Climbing' (HC) and for 'Simulated Annealing' (SA);
- ✓  $T_r = I+1$  for the 'Genetic Algorithm' (GA) under 'Improve', 'Replacement', and 'Keep the best' selection schemes;
- ✓  $T_r = I*2+1$  for the 'Genetic Algorithm' (GA) under 'Tournament', and 'Rank' selection schemes;
- ✓  $T_r = P*I+1$  for the 'Metapopulation Genetic Algorithm' (MetaGA) under 'Improve', 'Replacement', and 'Keep the best' selection schemes;
- ✓  $T_r = (P+1)(I+1)$  for the 'Metapopulation Genetic Algorithm' (MetaGA) under 'Tournament', and 'Rank' selection schemes with one CPU core;
- ✓  $T_r = (2P)(I+1)$  for the 'Metapopulation Genetic Algorithm' (MetaGA) under 'Tournament', and 'Rank' selection schemes with more than one CPU core;

$P$  is the number of populations and  $I$  is the number of individuals per populations.

For example, using a computer with 4 CPU cores, and using the metaGA (with 'Improve' Selection) with 4 populations of 4 individuals, and rate heterogeneity with 4 Gamma-rate categories on a DNA dataset of 120 taxa and 4000 nucleotides (hence, about 2500 data patterns, although that number can vary, depending on each specific dataset), will require about:

- a. **2.4 Gb** of RAM for a single core assigned to each replicate but 4 cores assigned to 4 simultaneous replicates;
- b. **1.2 Gb** of RAM for 2 cores assigned to each replicate and 2 cores assigned to 2 simultaneous replicates.

Note that option *a.* will be significantly faster than option *b.* Also note that:

- ✓ The amount of RAM computed above is a lower bound as the storage of the dataset itself can take a few hundreds Mb;
- ✓ An estimate of the amount of RAM necessary for your analysis is indicated in the parameter summary panel of the main window (Fig. 2) as well as in the lower-left corner of the 'Analysis settings' window (Fig. 9 to 18), on the basis of the parameters you have chosen in that same

<sup>2</sup> A data pattern is an aligned column with a specific combination of states. One pattern can occur several times within the same dataset. For example, the character columns 1, 8 & 9 below are identical, hence, their likelihoods are identical and must be computed only once (but used three times for computing the joint likelihood). Similarly, characters 3 & 7 are identical. The example dataset below exhibits 9 characters but only 5 patterns. The number of data patterns is indicated in the 'MetaPIGA data matrix' tab (see Fig. 3)

Character-->	1	2	3	4	5	6	7	8	9		Pattern -->	1	2	3	4	5
Taxon1	A	G	T	G	C	C	T	A	A		Taxon1	A	G	T	G	C
Taxon2	A	G	T	G	C	C	T	A	A		Taxon2	A	G	T	G	C
Taxon3	T	T	T	G	C	C	T	T	T	-> Compress ->	Taxon3	T	T	T	G	C
Taxon4	T	T	T	G	C	C	T	T	T		Taxon4	T	T	T	G	C
Taxon5	T	-	T	G	C	C	T	T	T		Taxon5	T	-	T	G	C
Pattern -->	1	2	3	4	5	5	3	1	1		Weight -->	3	1	2	1	2



window. In both windows, the estimate turns red if you exceed the amount of memory you allocated to MetaPIGA.

As indicated in Figure 1a, you can choose the amount of RAM assigned to MetaPIGA in the menu: *'Tools' → 'Memory Settings'*. You will be prompted by the program to do so if you experience an out-of-memory error during the use of MetaPIGA. The amounts of memory assigned, used, and available can be found in the menu *'Help' → 'System informations'* (Fig. 1b).

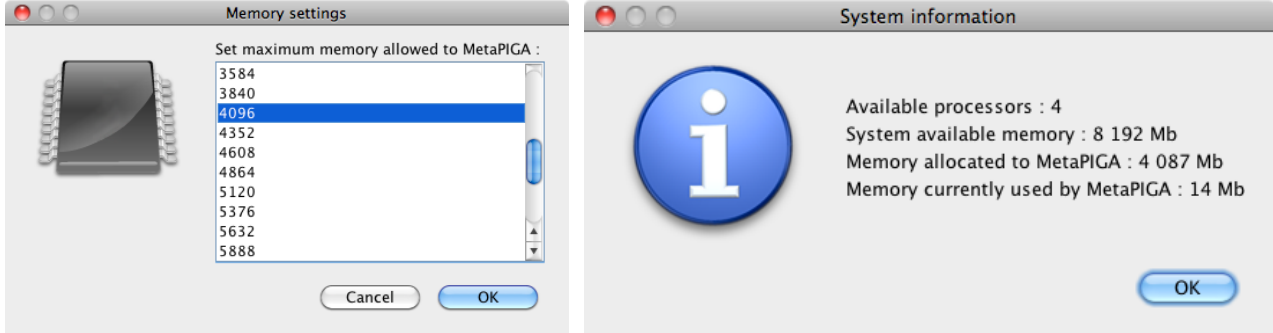


Fig. 1: The metaPIGA (a) Memory Settings and (b) System Information windows

**Graphics card memory.** For best performances, the graphics card must have enough built-in memory. To calculate the minimum amount of memory in megabytes, use the following formula:

$$RAM_{opt}^{GPU} (Mb) = \frac{12 \cdot C \cdot D \cdot S + 8 \cdot C \cdot D + 16 \cdot C \cdot S^2 + 8 \cdot C + 12 \cdot D + 16 \cdot S^2 + 16 \cdot S}{1024^2} .$$

Where D is the number of data patterns (see above), C is the number of discrete categories of the gamma distribution (typically, 4), and S is the number of possible character states (S=4, S=20, and S=64 for DNA, protein, and codon sequences respectively).

If the amount of available memory is less than that computed above, MetaPIGA will have to split the data into pieces before sending it to the GPU, which in turn degrades the performances of the GPU. To calculate the minimum of built-in GPU memory needed, use the following formula:

$$RAM_{min}^{GPU} (Mb) = \frac{8 \cdot C + 12 \cdot D + 8 \cdot C \cdot D + 16 \cdot S + 384 \cdot C \cdot S + 16 \cdot S^2 + 16 \cdot C \cdot S^2}{1024^2} .$$

## 5. Using MetaPIGA

### 5.1. Summary

MetaPIGA uses standard formats: reading and writing datasets in Nexus format [40] and trees in [Newick format](#). Note that aligned datasets in Fasta format can also be imported in MetaPIGA. All search settings can be saved in a metaPIGA block incorporated into the Nexus file, allowing easy management and runs on distant servers. A Nexus file without a metaPIGA block will be correctly interpreted by MetaPIGA and will run with default parameters (but it will skip other programs blocks such as ‘*Paup*’ or ‘*Assumptions*’ blocks). Note that the command “Endblock” often used in Paup data files is not a standard Nexus command and will not be recognized by MetaPIGA (please, use the standard Nexus command “END” instead). The minimum requirements are a *DATA* block (defining the datatype, the number of taxa and the number of characters), including a *MATRIX* command (*i.e.*, with the sequence data; if the matrix is in interleaved form, please, indicate it in the *DATA* block) with each sequence beginning with the sequence name separated from the sequence itself by at least one space. Standard ambiguity characters are accepted (see below) and missing data (defined by the ‘*MISSING*’ command; default is ‘?’) are automatically converted to ‘*N*’ (nucleotide sequences) or ‘*X*’ (amino-acid sequences). Gaps (defined by the ‘*GAP*’ command; default is ‘-’) can be removed (with the corresponding character in other taxa) or treated as ‘*N*’ (see *Section 5.3*).

<p><b>Example of Nexus file with nucleotide data.</b></p> <pre>#NEXUS BEGIN DATA;     DIMENSIONS NTAX=5 NCHAR=12;     FORMAT DATATYPE=DNA interleave     MISSING=? GAP=- ; MATRIX mysequence_T1 AGTGCCTGATTG mysequence_T2 AGTGCCTGATCG mysequence_T3 TTTGCCTG---G mysequence_T4 TTTGCCTAATCG mysequence_T5 T-TGCCTAATCG ; END;</pre>	<p><b>The standard ambiguity code for DNA sequences.</b></p> <p><b>M</b> = A or C  <b>V</b> = A or C or G (not T)  <b>R</b> = A or G  <b>H</b> = A or C or T (not G)  <b>W</b> = A or T  <b>D</b> = A or G or T (not C)  <b>S</b> = C or G  <b>B</b> = C or G or T (not A)  <b>N</b> = A or C or G or T</p>
<p><b>Example of Nexus file with protein data.</b></p> <pre>#NEXUS BEGIN DATA;     DIMENSIONS NTAX=5 NCHAR=12;     FORMAT DATATYPE=PROTEIN interleave     MISSING=? GAP=- ; MATRIX mysequence_S1 QSGT mysequence_S2 RSGT mysequence_S3 P-GK mysequence_S4 RLK mysequence_S5 RLK- ; END;</pre>	<p><b>The standard ambiguity code for PROTEIN sequences.</b></p> <p><b>B</b> = N or D  <b>Z</b> = Q or E  <b>J</b> = I or L  <b>X</b> = any amino-acid</p>

MetaPIGA can be run in command line (cf. end of ‘*Section 5.2*’, then jump directly to *Sections 5.7* and *5.8* as well as *Appendix 2*), but it also offers an extensive graphical user interface (GUI) for access to:

- ✓ Dataset setting (Fig. 4-9) : defining and managing charsets; including/excluding taxa, characters, and charsets; defining and managing dataset partitions; changing nucleotide sequences to codon sequences and vice versa;

- ✓ Analysis settings (Fig. 10-18): choosing and customizing heuristics; defining substitution models and their parameters; choosing starting-tree options; controlling operators; defining stop criteria and replicates, managing multi-core processing.

All settings are associated with an **interactive ‘mouse-over’ help system** such that, if you are an experienced phylogeneticist, you probably don’t need this manual much ;).

MetaPIGA implements simple **dataset quality controls** (testing for the presence of identical sequences and for excessively ambiguous or excessively divergent sequences) and automated trimming of poorly aligned regions using the *trimAl* algorithm [2]. MetaPIGA also implements **statistical methods for selecting substitution models that best fits the data** ([41]; and refs therein): the *Likelihood Ratio Test*, the *Akaike Information Criterion*, and the *Bayesian Information Criterion*.


The MetaPIGA GUI provides a **detailed run window** showing graphs specific to the corresponding heuristic. For example,, for a metaGA search with replicates, the run window shows: (i) the current best likelihood progression of each population and (ii) the current topology, posterior probability values, and average branch lengths of the consensus tree.

**Batch files** are particularly useful for running sequentially a single data set under multiple different settings or several datasets with the same settings. MetaPIGA supports the use of batch files that can be either written manually (see *Section 5.8*) or generated using tools available in the GUI (see *Section 5.7*): datasets and their settings can be duplicated, settings can be “stamped” from one dataset to another, and multiple combinations of datasets and settings can be saved in a batch file that can be run either in the GUI (with various graphical information on search progress) or using command line.


Input and **result trees** are manipulated in Newick format, but visualized graphically in the GUI, and can be exported for other programs. MetaPIGA also integrates a **Tree Viewer** that allows viewing, re-rooting, and printing trees as well as computing the likelihood of any tree (under any available substitution model) and optimizing its model parameters. Five other tools are implemented: a **Tree Generator** (using the starting tree settings), an **Ancestral State Reconstruction viewer** (associated with the Tree Viewer), a **Consensus Builder** (using user-trees and/or trees saved in the ‘Tree Viewer’), a tool for computing **Pairwise Distances**, and a **Memory Settings** tool defining the maximum amount of memory allocated to the program. See section 5.9 for details.

## 5.2. Launching MetaPIGA & opening a file

### 5.2.1. Loading a file

Double-clicking a ‘.nex’ file (on Windows and Mac OS X) launches MetaPIGA and opens the *Nexus* file. If it does not, launch MetaPIGA by double-clicking the application icon and open your NEXUS (or FASTA) file by clicking on the ‘Load Nexus file’ button  (Figs. 2 & 3) or by selecting in the menu: ‘File’ → ‘Load a Data File (Nexus or Fasta format)’. Several Nexus files can be loaded sequentially using the Load Nexus File button/command but multiple files can also be dragged and dropped from the OS navigator to the left panel of the MetaPIGA main window (Fig. 2). The upper-right and lower-right panels of the main window indicate the parameters and the data matrix, respectively, obtained from the corresponding *Nexus/Fasta* file (Fig. 2). The entry window gives access to a second tab (arrow in Fig. 2) that shows the compressed data matrix and indicates the number of data patterns and base frequencies.

### 5.2.2. Data quality control & alignment trimming

Hitting the ‘scissor’ button  (Fig. 2) in the center of the main window will successively launch quality tests for:

- ✓ The presence of excessively ambiguous sequences: sequences with >40% ambiguities (gaps and N/X) will be detected and will be proposed to be automatically removed.

- ✓ The presence of redundant sequences: groups of identical sequences will be detected and only one sequence (with the lowest number of ambiguities) will be kept for each such group<sup>3</sup>.
- ✓ The presence of excessively divergent sequences: if sequences generating large uncorrected pairwise distances (85% for proteins, 65% for nucleotide data, and 45% for standard binary data) are detected, a warning is given, suggesting to remove these sequences and to subsequently realign the dataset. MetaPIGA does not perform alignment, so you'll have to realign your sequences using an alignment software such as ClustalW or CodonCode Aligner.
- ✓ Automated trimming of poorly aligned regions using the *trimAl* algorithm [2]: excessively gapped and/or divergent positions are put in a charset of excluded characters (but they can be easily re-included in the 'Dataset settings', see section 5.3).

Each of these 4 tests is also separately accessible in the 'dataset' menu.

The *trimAl* algorithm has not yet been implemented for codon sequences in the MetaPIGA version 3.0.

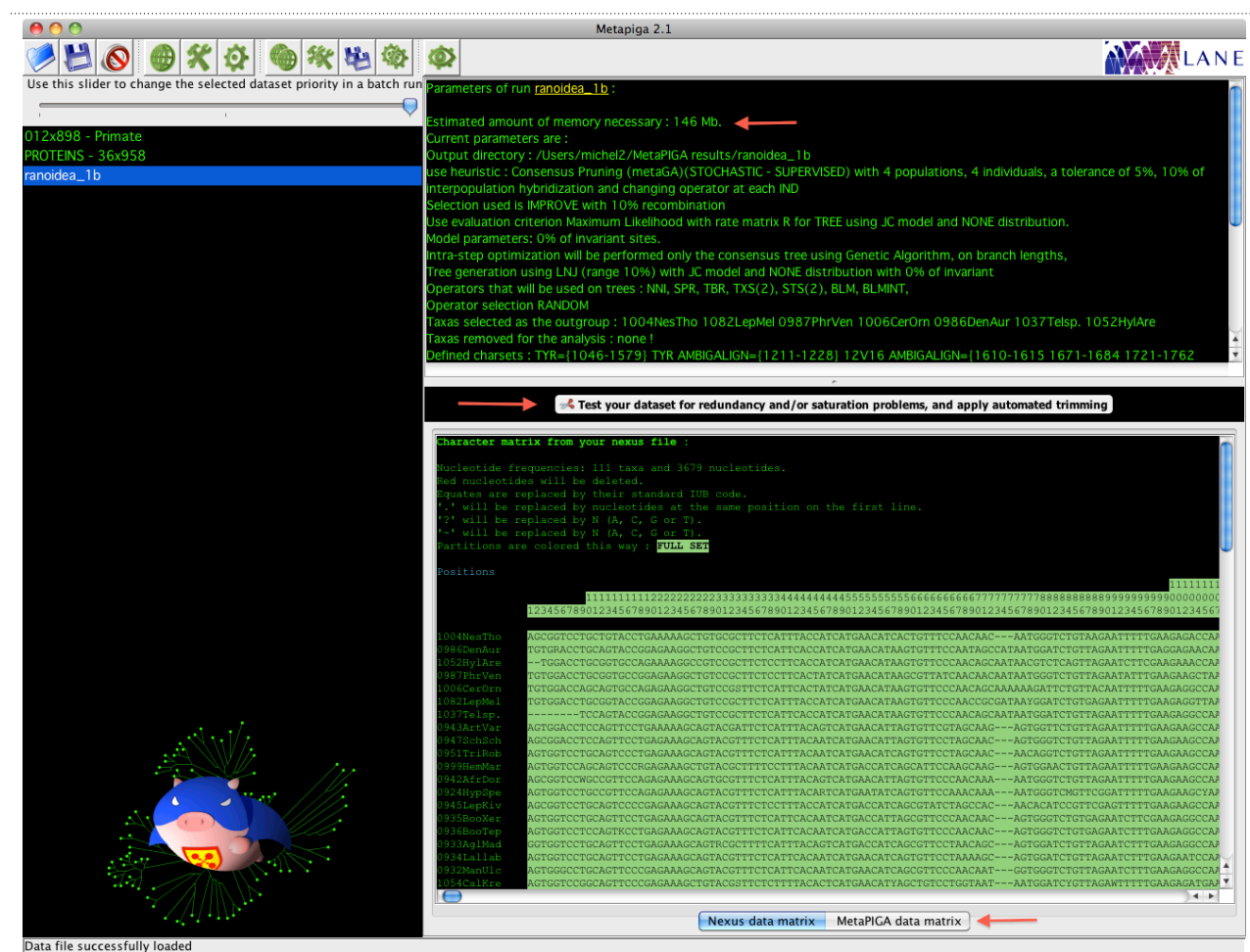


Fig. 2: The MetaPIGA main window with three loaded datasets and the 'ranoidea\_1b' dataset selected. The arrows indicate the memory required for running that dataset (under the current settings), the central button for data quality control & alignment trimming, and the second tab giving access to the compressed dataset, number of data patterns, and base frequencies.

<sup>3</sup> Ambiguities are considered as such during comparisons of sequences. For example, in the dataset below, there are two groups of identical sequences (seq.1+2+3 and seq.4+5). After running the test, MetaPIGA keeps, within each group, only the sequences with the lowest number of ambiguities (sequences 1 and 4).

Sequence1	A	G	T	G	C	C	N	G	A
Sequence2	A	G	Y	G	C	C	T	R	A
Sequence3	A	N	T	N	C	-	T	G	A
Sequence4	T	T	T	G	C	C	T	-	T
Sequence5	T	-	-	G	C	C	T	A	T

---->

Sequence1	A	G	T	G	C	C	N	G	A
Sequence4	T	T	T	G	C	C	T	-	T

The icons in the upper-left corner of the window (Fig. 3) are shortcuts to the main commands from the ‘File’, ‘Search’, ‘Batch’, and ‘Tools’ menus. Most of these functions are self-explanatory and are associated with an interactive ‘mouse-over’ help system. We will however discuss below the major functionalities. Most commands can be called using a short-cut of type ‘Ctrl/Cmd+letter’ (e.g., ‘Ctrl/Cmd+L’ for opening a Nexus or Fasta file).

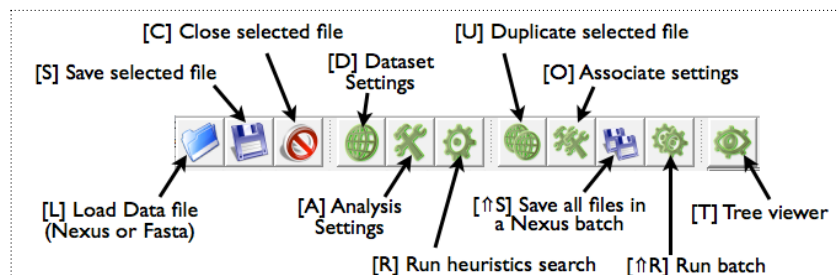


Fig. 3: The MetaPIGA main functionalities icons. These functionalities (and others) are also available through the ‘File’, ‘Dataset’, ‘Search’, ‘Batch’, and ‘Tools’ menus. Between brackets: shortcut command letters.

**NOTE: COMMAND LINE LAUNCH.** It is particularly useful to launch MetaPIGA in command line if you want to send jobs to a distant server. You must use the ‘mp\_console’ executable (and not ‘MetaPIGA’). Simply type the command “mp\_console” with the following arguments:


- ✓ **[noupdate]** : MetaPIGA will not check the MetaPIGA download server for an update;
- ✓ **[nogui]** : MetaPIGA will run without graphical interface (but textual progress), executing all files given in argument.
- ✓ **[width=]** : set the console width (default = 80). Necessary for progress bar display without GUI.
- ✓ **[silent]** Launches MetaPIGA without any GUI or text progress.
- ✓ **[aFilename]** : The Nexus/Fasta file that will be opened by MetaPIGA and executed if [nogui] is set. If several file-names are given, they will be run sequentially as a batch.

For example, to run sequentially two nexus files ‘file1.nex’ and ‘file2.nex’ without GUI under Windows, type: “mp\_console.exe noupdate nogui file1.nex file2.nex”

Refer to ‘Section 5.8’ on how building batch files manually, and to ‘Appendix 2’ for the full list of MetaPIGA commands that can be incorporated in Nexus files.

## 5.3. [D] Dataset Settings

### 5.3.1 Overview

The dataset settings are accessed by clicking on the button  or by selecting in the menu: ‘Dataset’ → ‘Dataset settings’. This window allows to:

- ✓ define and manage charsets;
- ✓ include/exclude taxa, characters, and charsets;
- ✓ define and manage dataset partitions;
- ✓ define outgroup sequences;
- ✓ define a range of Codons inside a nucleotide sequence.

This window is divided into two tabs. The first tab (*Dataset*) handles charsets, partitions, outgroups, and excluded taxa. The second (*Codons*) allows defining Codon characters in nucleotide sequences.

The corresponding window for the ‘ranoidea\_1b.nex’ file is shown below (Fig. 4). The 7 outgroup taxa and the 10 charsets were predefined (hence, recognized by the program) in the nexus file using a metaPIGA block as highlighted in green below. See Appendix 2 for the full list of MetaPIGA commands.

```
#NEXUS
BEGIN DATA;
DIMENSIONS NTAX=111 NCHAR=3679;
FORMAT DATATYPE=DNA interleave MISSING=? GAP=- ;
MATRIX
The data matrix is here in interleaved format
;
END;
BEGIN METAPIGA;
```



```

charset name=RAG1 set{1-555};
charset name=rhod1 set{556-870};
charset name=rhod4 set{871-1045};
charset name=Tyr set{1046-1579};
charset name=12V16 set{1580-3080};
charset name=16S set{3081-3679};
charset name=RAG_AmbigAlign set{67-84};
charset name=Tyr_AmbigAlign set{1211-1228};
charset name=12V16_Ambigalign set{1610-1615 1671-1684 1721-1762 1784-1801 1817-1867
1892-1900 1911-1915 1953-1999 2048-2055 2070-2086 2107-2116 2128-2199 2208-2219
2237-2262 2287-2304 2308 2324-2332 2349-2352 2363-2370 2378-2390 2411 2431-2454
2567-2611 2673-2700 2722-2728 2742-2831 2864-2884 2900-2988 3022-3080};
charset name=16S_AmbigAlign set{3090-3103 3128-3136 3149-3158 3318-3398 3438-3507
3513-3527 3649-3679};
outgroup {1004NesTho 0986DenAur 1052HylAre 0987PhrVen 1006CerOrn 1082LepMel 1037Telsp.};
end;

```

All commands can be performed with the GUI (instead of using commands in the Nexus file) as described below.

### 5.3.2 The 'Dataset' tab

Use the >> and << buttons to (i) transfer taxa in and out of the outgroup, (ii) exclude/include taxa, (iii) consider/disregard pre-defined charsets as partitions, and (iv) include/exclude character sets from the analysis. Character sets ('charsets') can be defined and managed using the interface (see below). In Fig 4, one taxon and four charsets were excluded manually. The 'Charset viewer' button allows selecting and visualizing any of the charsets (highlighted in the full dataset). Clicking on the 'Define new charset' button

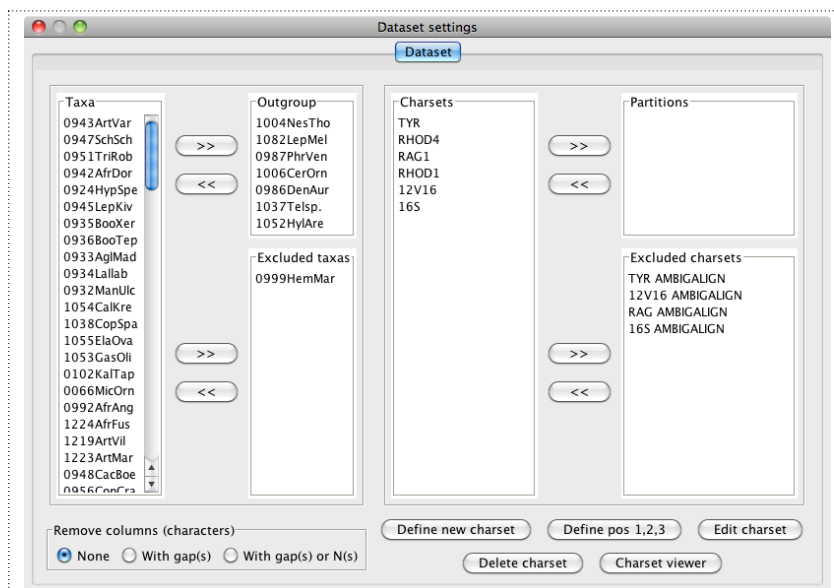


Fig. 4: The 'Dataset settings' window.

opens a window for selecting characters to include in the new charset. Multiple selections can be performed with the mouse (and shift/ctrl/cmd keys depending on your OS) or a range selection tool.

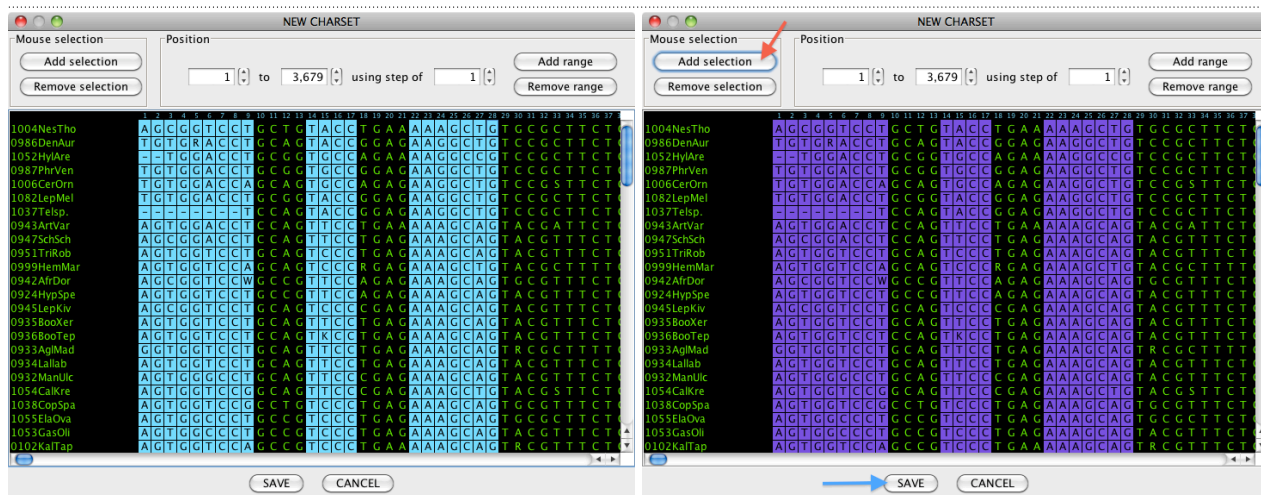


Fig. 5: The selection tool for defining new character sets. Select the characters to be included in the charset and click



In the first example (Fig. 5), a set of 9+4+7 characters have first been selected with the mouse, then added to the *charset* under construction by using the ‘**Add selection**’ button (red arrow in Fig. 5). That button can be used multiple times to sequentially add different sets of characters to your new *charset*. Once a selection has been added, its colour is changed to avoid any ambiguity. Once clicking the ‘**SAVE**’ button, you will have to supply a name (here, we used ‘**MYCHARSET**’) for the new *charset* and it will appear in the list of available *charsets* (Fig. 7).

Note that a charset can also be selected using the *range selection* tool as in this second example (Fig. 6) where nucleotides between position 1 and 300 have been selected every three positions. This allows for example to easily define 1<sup>st</sup>, 2<sup>nd</sup>, and 3<sup>d</sup> positions in a protein-coding sequence. The *mouse selection* tool (Fig. 5) and the *range-selection* tool (Fig. 6) can be used in combination. If your dataset is exclusively made of in-frame protein-coding nucleotide sequences, quick definition of first, second, and third positions can be performed using the ad-hoc ‘*Define pos 1,2,3*’ button in the ‘Dataset settings’ (Fig. 4).

Charsets can then be excluded/included from the analysis or considered/disregarded for data partitioning. In the example in Fig. 7, we have 7 taxa in the outgroup, 1 excluded taxon, 11 charsets of which 4 are excluded (in the present case, these are ambiguously aligned positions for different genes, hence, it was chosen to remove them from the analysis), and 3 partitions: ‘16S’, ‘MY\_CHARSET’, All other non-excluded characters (automatically grouped into a virtual charset named “REMAINING”)<sup>4</sup>.

**Gaps.** The user can choose to remove, before the analysis is performed, either all columns with at least one gap, or at least one gap or one ‘N’ (‘A’ or ‘C’ or ‘G’ or ‘T’).

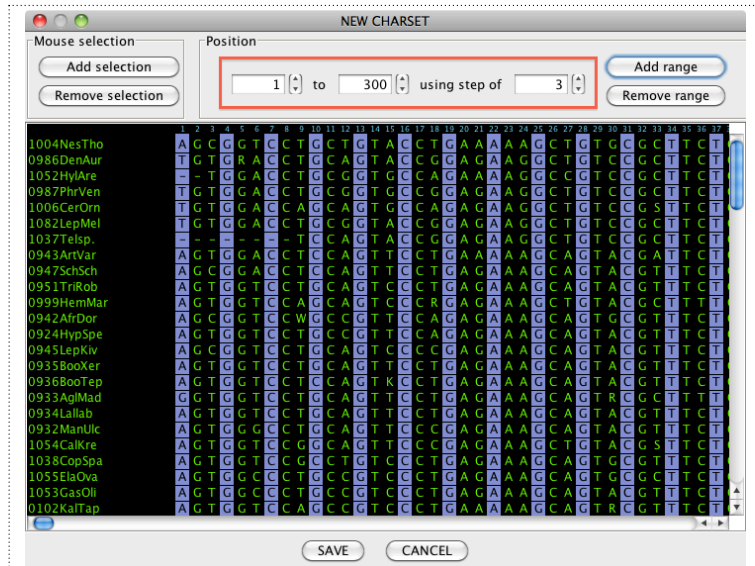


Fig. 6: Defining a character set with the range-selection tool.

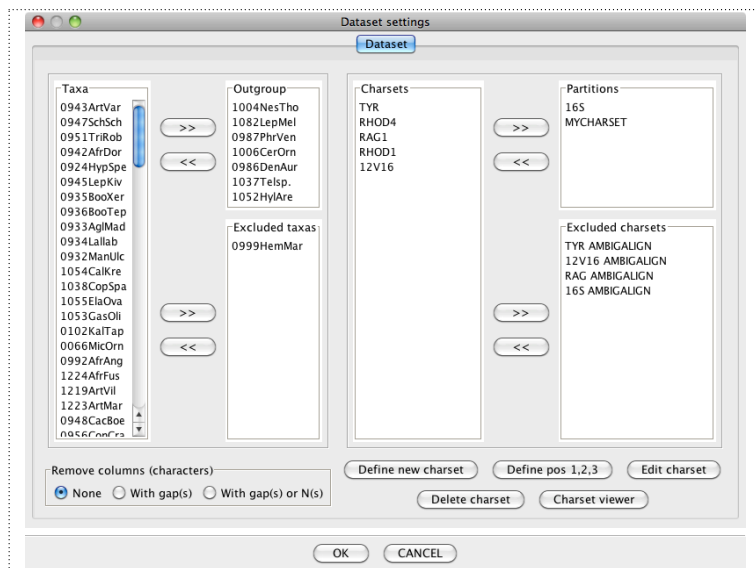


Fig. 7: The ‘Dataset’ window after defining the new charset (‘MY\_CHARSET’) and partitioning of the data.

<sup>4</sup>We assume that all partitions evolve on the same topology, but all other parameters (base freq, substitution matrix rates, shape parameter of  $\gamma$ -*distr*, and proportion of invariable sites *Pinv*) are estimated and optimized separately for each partition. Among-partition rate variation parameters are introduced in the likelihood equation as a factor that modifies branch lengths for the corresponding partition. Branch lengths are optimized as usual, but the relative rates of partitions are optimized separately (with the constraint that the weighted average of among-partitions rates is 1; weighting is according to each partition’s size). See Appendix 3 for details.

### 5.3.3 The ‘Codons’ tab

The codon tab consists of a *codon range viewer* and two buttons that are used for codon range definition. Codons are indicated with black letters on a light green background. The remaining of the dataset is colored inversely (Fig. 8). Pressing the ‘Make codons’ button will open the

*codon maker window* where you can define (i) the range of the coding sequence and (ii) the genetic code you wish to use (i.e., The Universal Code, the Vertebrate Mitochondrial Code, etc., see below). The range of coding sequences can be defined by manually picking the first position in the dataset and pressing the ‘Set as first position’ button in the upper left corner of the window (Fig. 9, highlight a). Similarly, pick the last position in the dataset and press the ‘Set as last position’ button. Alternatively, define the range by entering the indexes of the first and the last positions in the top middle part of the window (Fig. 9, highlight b). Note that the first and last positions must define a range corresponding to a multiple of 3 nucleotides. If this

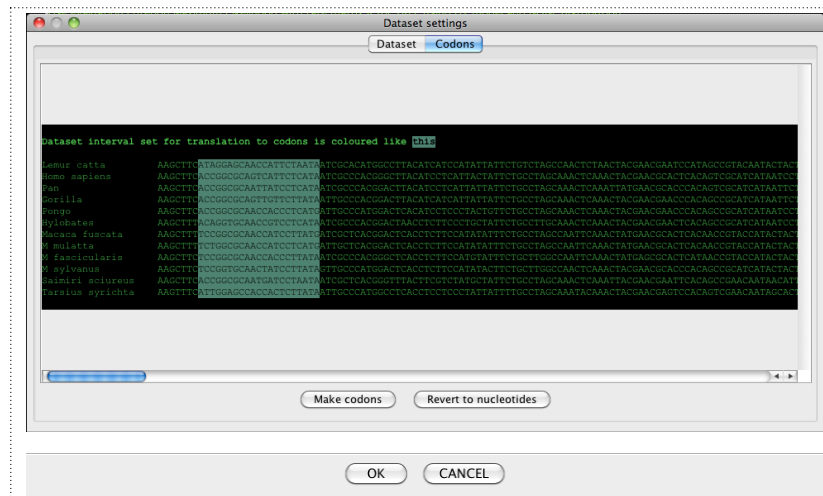


Fig. 8: The ‘Codons’ tab after defining the codon range within the sequence. The Codon range is marked with the green background.

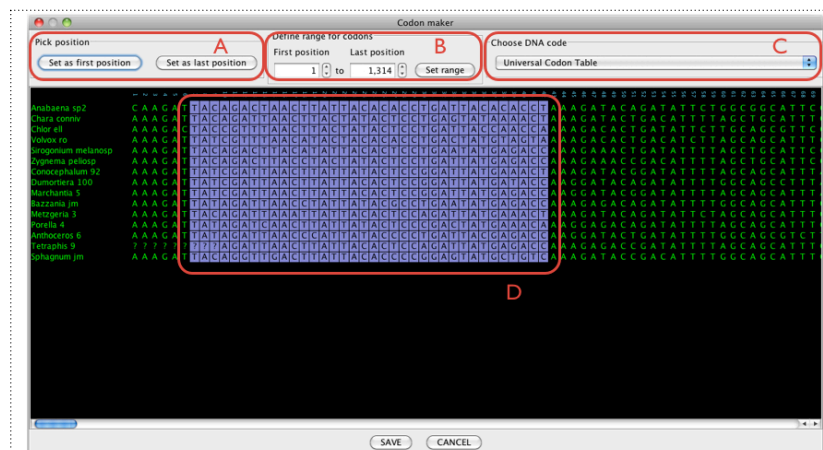


Fig. 9: The codon maker. Tools for defining codon range (a and b), the drop-down menu for selecting a DNA code (c), and the range of nucleotides selected as codons (d, in purple) are indicated.

is not the case,, the codon maker will trim the range to the closest smaller third nucleotide position. Also, note that if some of the codons are either stop codons or ambiguous codons, the codon maker will exclude the corresponding codons and a warning will pop up. Important: The nucleotides outside of the defined range of codons will be ignored during subsequent analyses. If you have charsets defined before the translation to the codons, these charset will be available only if they are compatible with the codon range. These incompatible charsets will become available again as soon as you revert to the nucleotide character mode (see below). If you are saving a codon range to a nexus file, the incompatible charsets will not be saved.

The genetic codes (for codon translation) available in the drop-down menu (Fig. 9c) are:

- ✓ The Universal Code;
- ✓ The Ciliate, Dasycladacean and Hexamita Nuclear Code;
- ✓ The Echinoderm and Flatworm Mitochondrial Code;
- ✓ The Euplotid Nuclear Code;
- ✓ The Invertebrate Mitochondrial Code;

- ✓ the Mold, Protozoan, Coelenterate Mitochondrial & The Mycoplasma/Spiroplasma Code;
- ✓ The Vertebrate Mitochondrial Code.

For additional information on genetic codes, please check: <http://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi>.

Once you have defined the codon range and the genetic code, press the 'Save' button and proceed to defining your Analysis Settings (section 5.4 below).

**Revert to nucleotides.** To revert defined codons back into nucleotide characters, please press the 'Revert to nucleotides' button (Fig. 8). If some of your charsets became unavailable during codon definitions, they will re-appear in the list of charsets.

### 5.3.4 Exiting the Settings Window

Once outgroup sequences, charsets, partitions, and excluded sequences and charsets have been defined (and, potentially, the range of codons), and the 'OK' button has been hit, the main (entry) window is updated (Fig. 10): the upper-right window lists the new settings and the lower-right window indicates the excluded characters and excluded taxa in red, and the various partitions using a color-coded font background. Switching to another dataset in the left window and modifying the settings for that dataset does not affect the settings associates to the other datasets.

**Note:** A dataset can be saved as a Nexus file with both excluded taxa and excluded charset deleted from the DATA matrix. To do this, use the menu 'File > Save modified dataset to Nexus'.

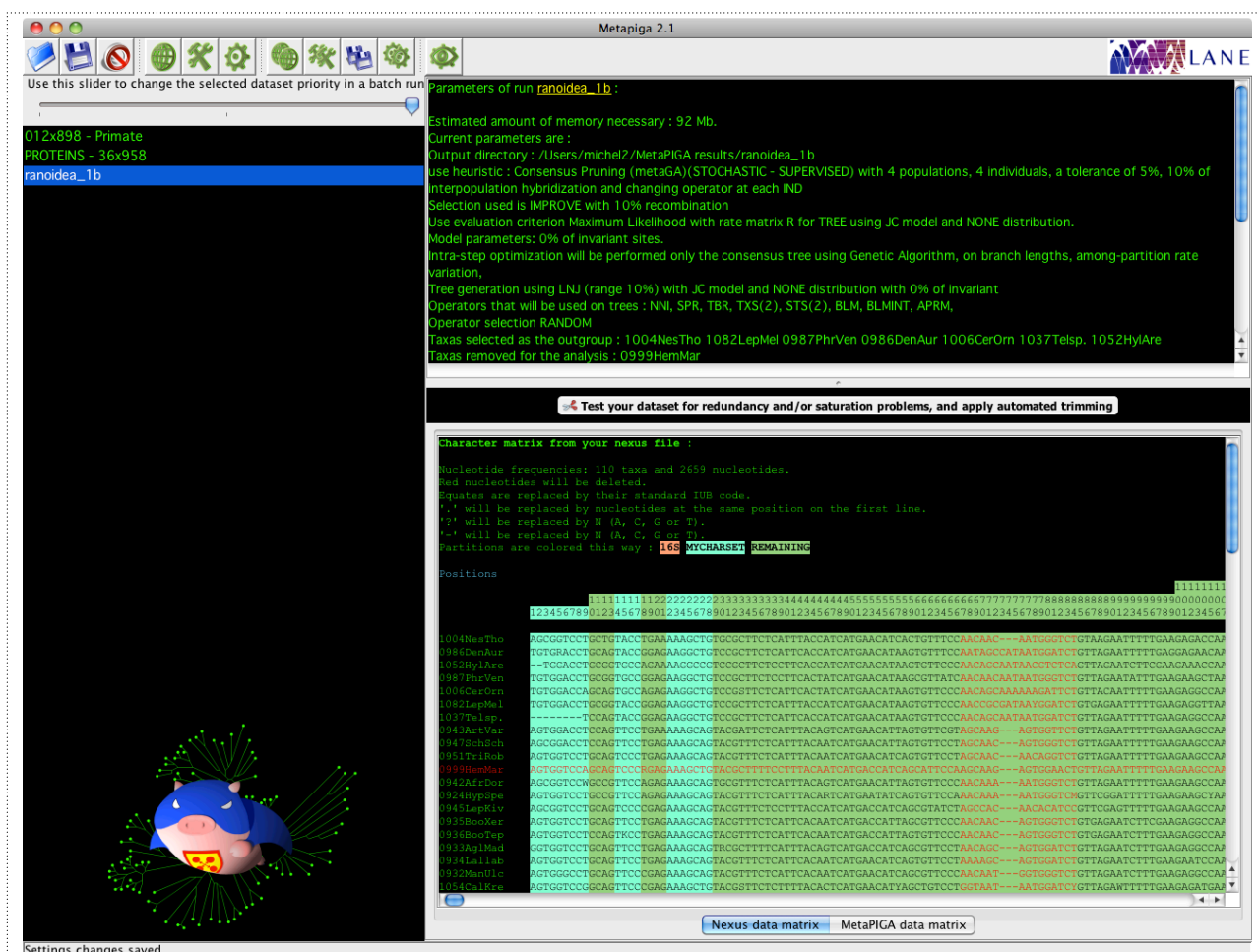



Fig. 10: The MetaPIGA entry window updated after defining the settings.

## 5.4. [A] Analysis Settings

The analysis settings are accessed by clicking on the button  or by selecting in the menu: 'Search' → 'Analysis settings'. The *Settings* window includes 5 tabs to switch among the corresponding parameter controls relevant to: '**Heuristic**', '**Evaluation Criterion**', '**Starting tree(s)**', '**Operators**', and '**Miscellaneous**'. The user can switch from one tab to another and confirm ALL changes by clicking on the 'OK' button in ANY of the tabs.

Note that the analysis settings window always indicates in the lower left corner (blue frame in Fig. 11) the amount of memory necessary for running the analysis given the settings so far selected. When the amount of memory exceeds that allocated to MetaPIGA, the corresponding text turns red. To alleviate the problem, use 'Tools' → 'Memory Settings' (Fig. 1) to increase the memory allocated to MetaPIGA.

### 5.4.1. The 'Heuristic' tab

We implemented four heuristics in MetaPIGA: a *hill climbing* algorithm, a *Stochastic Simulated Annealing* algorithm (SSA; [26, 42]), a classical *Genetic Algorithm* (GA; [27-29]), and the *meta-population Genetic Algorithm* based on the *Consensus Pruning* principle (metaGA; [1]), all available in the *Heuristic tab* (Fig. 11).

#### The Hill Climbing (HC) algorithms

The '*Stochastic HC*' algorithm generates a new solution tree at each step (using available operators) and accepts it only if its likelihood is better than the current solution. HC algorithms are fast but tend to generate solutions trapped in local optima and are therefore highly dependent on the starting tree localization in tree space as well as on the (unknown) tree space topography.

Hence, the user can choose to perform '*Random-restart hill climbing*' *i.e.*, an algorithm that iteratively performs  $N$  hill climbings, each time with a different initial tree. Among the  $N$  solution trees, only the best is kept. The user can fix the number of restarts (20 by default).

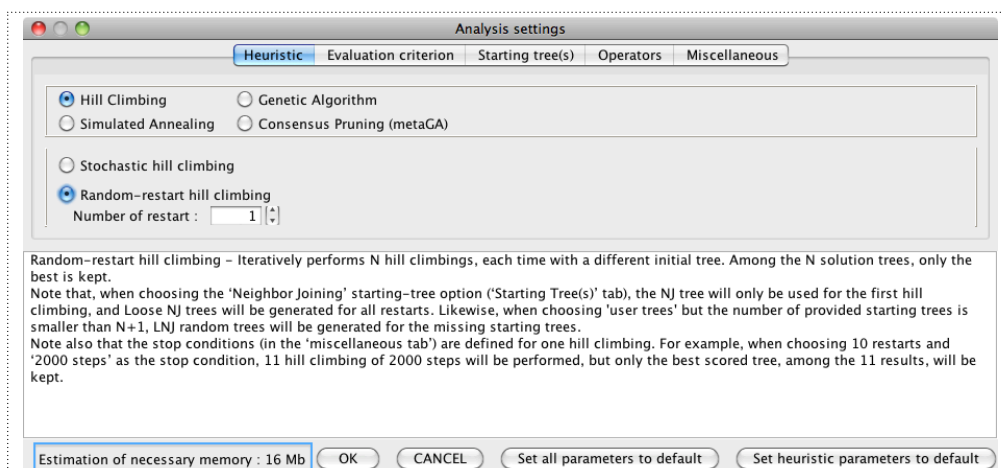


Fig. 11: The '*Heuristic*' window with the '*Hill Climbing*' heuristic selected and the corresponding mouse-over help text. The blue frame highlights the amount of memory required for running the analysis given the settings so far selected.

Figure 11 also illustrates the 'mouse-over' help system of MetaPIGA: an explanatory note appears when moving the mouse cursor over the corresponding field, parameter, or radio-button, etc. In figure 11, the mouse cursor is over the '*Random-restart Hill Climbing*' radio button.

#### The Stochastic Simulated Annealing algorithm (SSA)

The SSA algorithm uses statistical mechanics principles to solve combinatorial optimization problems [42]; *i.e.*, it mimics the process of minimal energy annealing in solids. The first attempt to use this approach for the evolutionary tree problem was introduced in 1985 by Lundy [43], and its use for ML phylogeny inference was further developed in 2001 by Salter and Pearl [26]. SSA starts with an initial state (the starting tree) and randomly perturbs that solution (using available tree op-



erators). If the new state is better (lower energy, better likelihood), it is kept as the new current state; if the new state is worse (higher energy, worse likelihood), it is accepted as the current state with the probability  $e^{\Delta E/T}$ , where  $\Delta E$  is the negative difference in energy (here, the difference of likelihood) between the two states, and  $T$  is the so-called ‘temperature’ of the system. If  $T$  is lowered slowly enough, the algorithm is guaranteed to find the optimal solution, but if the temperature is lowered too slowly, the time to find the optimal solution can exceed that of an exact search. The obvious asset of the algorithm is its ability to momentarily accept suboptimal solutions, allowing it to escape local optima whereas its obvious drawback is the difficulty to define the shape and speed of the ‘cooling schedule’ (*i.e.*, the rate of the decrease in  $T$ ). Efficient schedules highly depend on the dataset. The efficiency of the algorithm is unknown and optimization of its parameters has never been performed. Before this optimization analysis (in progress) is finalized, the SSA is provided *as is* for allowing users to explore its utility. The parameters available in MetaPIGA 3 for the SSA are described in Appendix 2.

### The Genetic Algorithm (GA)

The GA is an evolutionary computation approach that implements a set of operators mimicking processes of biological evolution such as mutation, recombination, selection, and reproduction (*e.g.*, [44]). After an initial step of generating a population of trees, the individuals (specific trees with their model parameters) within that population are (i) subjected to mutation (a stochastic alteration of topology, branch lengths or model parameters) and/or recombination, and (ii) allowed to reproduce with a probability that is a function of their relative fitness value (here, their likelihood). Because selection preferentially retains changes that improve the likelihood, the mean score of the population improves across generations. However, because sub-optimal solutions can survive in the population (with probabilities that depend on the selection scheme), the GA allows, in principle, escaping local optima. In MetaPIGA, we implemented **5 alternative selection schemes** (Fig. 12, see [1]):

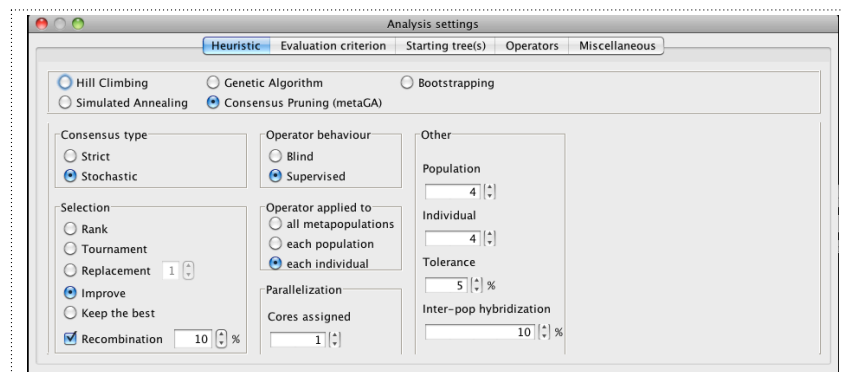


Fig. 12: The ‘Heuristic’ window with ‘Genetic Algorithm’ selected.

- ✓ ‘Rank’: individuals are assigned a probability of leaving an offspring (*i.e.*, a copy of themselves) as a function of their position in a list in which they are ranked by their score. The probability for the  $i^{\text{th}}$  individual of leaving an offspring to the next generation is equal to:

$$\frac{2}{n(n+1)}(n-i+1)$$

- ✓ ‘Tournament’: two individuals are drawn randomly from the population of  $I$  individuals and one offspring is produced from the individual with the higher score. Both trees are then placed back into the mating population and the whole process is repeated until  $I$  offspring have been generated. This is the default selection scheme when using the GA.
- ✓ ‘Replacement’: two individuals are drawn randomly from the population of  $I$  individuals and two copies of the better individual are returned to the mating pool (parents are discarded). The process is repeated  $sI$  times, where  $s$  is the selection strength. The offspring population is generated as a copy of the post-selection parent population.
- ✓ ‘Improve’: only those individuals that have improved (in comparison to their likelihood at the previous generation) are allowed to produce an offspring. Each individual that fails this test is discarded and replaced by a copy of the current best individual.
- ✓ ‘Keep the Best’: only the best individual (*i.e.*, with highest likelihood) is kept and all other individuals are replaced by a copy of the best individual.

All selection regimes (except ‘*Improve*’ and ‘*Keep the best*’) tolerate the maintenance of poor trees in the evolving populations, an effect which allows escaping from local optima but increases search time (see below how the *metaGA* resolves that problem).

We also implement one *recombination scheme* where each sub-optimal individual has a probability (determined by the user) to recombine with a better individual. Recombination is performed by exchanging subtrees defined by one (if any) of the identical taxa partitions in the two parental trees (*i.e.*, one internal branch that defines subtrees including the same taxa but with potentially different sub-topologies). A recombination can be viewed as a large number of simultaneous topological mutations.

Beside the selection scheme and the possibility to perform intra-population recombinations, the major parameter in the GA is the **population size** (set by the user).

### The metapopulation Genetic Algorithm (*metaGA*)

This approach relies on the coexistence of  $P$  interacting populations [1] of  $I$  individuals each ( $P$  and  $I$  defined by the user): the populations are not fully independent as they cooperate in the search for optimal solutions. Within each population, a classical GA is performed: trees are subjected to mutation events, evaluation, and selection (5 alternative selection schemes are available as in the GA above). However, all topological operators are guided through inter-population comparisons defined and controlled by ‘**Consensus Pruning**’ (CP; [1]): topological consensus among trees across populations defines the probability with which different portions of each tree are subjected to topological mutations (Fig. 13). These comparisons allow the dynamic differentiation between internal branches that are likely correct (hence, that should be changed with low probability) and those that are likely incorrect (hence, that should be modified with high probability).

Although CP allows for many alternative inter-population communication procedures, we implemented (Fig. 14) the two that we identified as the most useful:

- ✓ ‘**Strict CP**’: internal branches shared by all trees across all populations cannot be affected by topological mutations, all other internal branches are unconstrained.
- ✓ ‘**Stochastic CP**’ (default): topological mutations affecting a given branch are rejected with a probability proportional to the percentage of trees across all populations that agree on that branch.

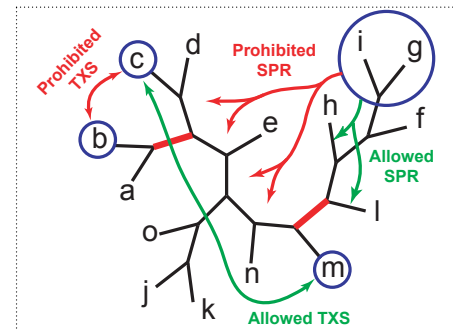


Fig. 13: The principle of CP. Before a tree is mutated, its topology is compared with those of the best trees from other populations; the consensus branches (bold red) define the partitions that can (green arrows) and cannot (red arrows) be affected by topological mutations; *i.e.*, any operation moving a taxon across a consensus branch is prohibited.

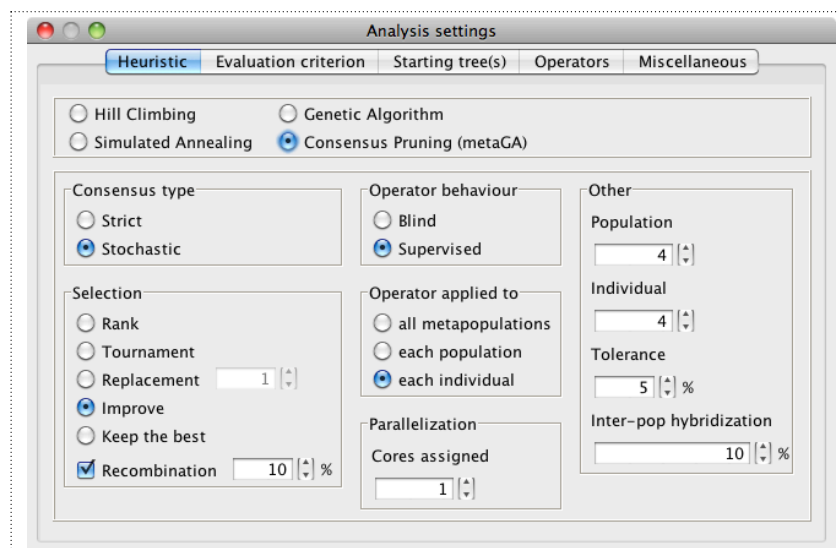


Fig. 14: The ‘Heuristic’ window with the ‘*metaPopulation Genetic Algorithm*’ selected.

**The default selection method** for the MetaGA is ‘*Improve*’ (see above). This scheme greatly re-



duces the intra-population variability after each selection step but local optima are avoided thanks to ‘*Consensus Pruning*’. In other words, the metaGA resolves the major problem inherent to classical Genetic Algorithms by maintaining high (inter-population) variation even under strong (intra-population) selection.

As constraining entirely an internal branch from being affected by topological mutations necessarily increases the likelihood to be trapped in a local optimum, a tolerance parameter  $t$  (defined to 5% by the user in Fig. 14) is implemented, allowing any internal branch to be affected with a probability  $t$  even if the corresponding branch is shared by all trees. The user of MetaPIGA has the choice between a ‘*blind*’ and a ‘*supervised*’ procedure for handling constrained partitions (Fig. 14). In the former, a topological mutation that affects a constrained branch is simply aborted and the tree is left unchanged, whereas in the latter, topological operators exclusively target branches in a pool of acceptable (unconstrained) candidates. The ‘*supervised*’ procedure is used as default because preliminary analyses suggest that it allows trees to converge faster to higher likelihoods.

The MetaGA allows for two, non-mutually exclusive, recombination flavors: ‘*intra-population recombination*’ (lower-left field in Fig. 15) where each sub-optimal individual at each generation has a probability (instead of being mutated) to recombine with a better individual from that population (as in the GA above), and ‘*inter-population hybridization*’ (lower-right field in Fig. 14) where, at each generation, there is a probability (defined by the user) that all sub-optimal individuals from one random population, instead of being mutated, are recombined with one individual from another population; sub-optimal individuals from other populations experience the normal mutation procedure.

As CP provides frequencies of internal branches shared among trees across populations, it also indicates if the populations converge towards a stable set of solutions, *i.e.*, towards a consensus with stable branch frequencies. Hence, CP provides a **stopping rule** not available to other heuristics: the user can choose to stop the search when a series of successive mean relative error (MRE) values remains below a threshold defined by the user. To increase independence among samples, MRE are computed every  $n > 1$  (*i.e.*, non-successive) generations. The user defines  $n$ , as well as for how many samples the MRE must remain below the specified threshold before the search stops. See Section 5.4.5 (The ‘*Miscellaneous*’ tab) for details.

#### 5.4.2. The ‘Evaluation criterion’ tab

##### Setting ML Models

This window allows defining substitution models and their parameters (Fig. 15). Trees are estimated in MetaPIGA with the Maximum Likelihood criterion (ML) using one of 5 nucleotide substitution models for DNA sequences, one of 11 amino-acid substitution models, or one of two codon models. The implemented nucleotide substitution models are ([3] and refs therein): ‘*Jukes Cantor*’ (JC), ‘*Kimura’s 2 parameters*’ (K2P), ‘*Hasegawa-Kishino-Yano 1985*’ (HKY85), ‘*Tamura-Nei 1993*’ (TN93), and ‘*General Time Reversible* (GTR)’. The available amino-acid substitution models are: the ‘*Poisson*’ and ‘*GTR20*’ models (extensions of, respectively, the JC and GTR models to the 20 by 20 substitution matrix of protein sequences), and 9 empirical models for mitochondrial, chloroplastic, and nuclear Protein sequences: ‘*MtMam*’, ‘*MtRev*’, ‘*RtRev*’, ‘*CpRev*’, ‘*BLOS-SUM62*’, ‘*VT*’, ‘*Dayhoff*’, ‘*JTT*’, and ‘*WAG*’. The implemented codon substitution models are *GY* and *ECM* (Empirical Codon Model) [45, 46]. For the empirical protein and codon models, state frequencies can be set to the empirical values used by the authors who designed the corresponding model. Alternatively, state frequencies can be set to those observed in the dataset under analysis. Analyses can be performed with Rate Heterogeneity among sites using either a discrete ‘*Gamma distribution of rates*’ ( $\gamma$ -distr) [35, 36] or a ‘*Proportion of Invariant Sites*’ (*Pinv*) [37], or both ( $\gamma$ -distr + *Pinv*). All parameters of the model (transition/transversion ratio or components of the rate matrix, the shape parameter of the  $\gamma$ -distr, and *Pinv*) can be set by the user or estimated from a NJ tree (using the ‘*Estimate starting parameters*’ button, blue frame, Fig. 15).



Analysis settings

Heuristic Evaluation criterion Starting tree(s) Operators Miscellaneous

**Nucleotide substitution model**

Model selection

- ☐ GTR
- ☐ TN93
- ☐ HKY85
- ☒ K2P
- ☐ JC

Model testing

Likelihood Ratio Test

Akaike Information Criterion

Bayesian Information Criterion

Cores assigned : 4

**Model parameters**

Partition : 16S

Rate matrix

Ratio transition:transversion

1 : 4

Starting values

	A	C	G	T
A	-	1	0.5	1
C	1	-	1	0.5
G	0.5	1	-	1
T	1	0.5	1	-

Rate heterogeneity

- ☒ None
- ☐ Discrete Gamma
  - subsets 4
  - shape 1.0

Invariable sites

- ☒ None
- ☐ P-Invariant
  - 0 %

**Intra-step optimization**

- ☐ never
- ☒ consensus tree only
- ☐ at the end of (each) search
- ☐ stochastic : 0.5 %
- ☐ discrete : 200 steps

Algorithm

Genetic Algorithm

**Targets**

- ☒ Branch lengths
- ☒ Rate matrix parameters
- ☐ Shape of Gamma distribution
- ☐ Proportion of invariable sites
- ☒ Among-partition rate variation

Estimate starting parameters

Analysis settings

Heuristic Evaluation criterion Starting tree(s) Operators Miscellaneous

**Amino acid substitution model**

Model selection

- ☐ GTR20
- ☒ WAG
- ☐ JTT
- ☐ Dayhoff
- ☐ VT
- ☐ BLOSUM62
- ☐ CpRev
- ☐ MtRev
- ☐ RtRev
- ☐ MtMam
- ☐ Poisson

Model testing

Akaike Information Criterion

Bayesian Information Criterion

Cores assigned : 4

**Model parameters**

Partition : CHARSET1

Rate matrix

Starting values

	A	R	N	D	C	Q	E
A	-	1.75	1.62	2.35	3.26	2.89	5.03
R	1.75	-	2.02	0.47	1.68	9.64	1.4
N	1.62	2.02	-	17.25	0.84	4.9	3.01
D	2.35	0.47	17.25	-	0.1	1.96	19.62
C	3.26	1.68	0.84	0.1	-	0.31	0.07
Q	2.89	9.64	4.9	1.96	0.31	-	17.38
E	5.03	1.4	3.01	19.62	0.07	17.38	-
G	4.5	1.86	3.58	2.75	0.97	1.05	1.8
H	1.01	6.79	12.57	2.96	0.79	13.64	1.81
I	0.61	0.59	1.76	0.13	0.54	0.36	0.4

Equilibrium aa frequencies

- ☐ Equal aa frequencies
- ☒ Empirical aa frequencies
- ☐ Estimated aa frequencies

Rate heterogeneity

- ☒ None
- ☐ Discrete Gamma
  - subsets 4
  - shape 1.0

Invariable sites

- ☒ None
- ☐ P-Invariant
  - 0 %

**Intra-step optimization**

- ☐ never
- ☒ consensus tree only
- ☐ at the end of (each) search
- ☐ stochastic : 0.5 %
- ☐ discrete : 200 steps

Algorithm

Genetic Algorithm

**Targets**

- ☒ Branch lengths
- ☐ Rate matrix parameters
- ☐ Shape of Gamma distribution
- ☐ Proportion of invariable sites
- ☒ Among-partition rate variation

Estimate starting parameters

Amino acid substitution model

Model selection

- ☒ GTR20
- ☐ WAG
- ☐ JTT
- ☐ Dayhoff
- ☐ VT
- ☐ BLOSUM62
- ☐ CpRev
- ☐ MtRev
- ☐ RtRev
- ☐ MtMam
- ☐ Poisson

**Model parameters**

Partition : CHARSET1

Rate matrix

JTT

Fill R matrix

Starting values

	A	R	N	D	C	Q	E
A	-	1.7525...	1.6199...	2.3480...	3.2632...	2.8869...	5.0292...
R	1.7525...	-	2.0187...	0.4680...	1.6782...	9.6447...	1.3953...
N	1.6199...	2.0187...	-	17.251...	0.8428...	4.9046...	3.0095...
D	2.3480...	0.4680...	17.251...	-	0.0962...	1.9597...	19.617...
C	3.2632...	1.6782...	0.8428...	0.0962...	-	0.3139...	0.0678...
Q	2.8869...	9.6447...	4.9046...	1.9597...	0.3139...	-	17.378...
E	5.0292...	1.3953...	3.0095...	19.617...	0.0678...	17.378...	-
G	4.5013...	1.8576...	3.5762...	2.7502...	0.9744...	1.0486...	1.8038...

Fig. 15: The ML model window for DNA (top panel) and Protein (middle panel) datasets. The purple arrow indicates the drop-down menu for selecting the character set for which the settings are being defined: all charsets must be analyzed with the same single model (K2P and WAG models are selected in the examples shown), but the parameter values of the chosen model (e.g., the *transition:transversion* ratio for K2P or the *estimated aa frequencies* for WAG) can be different for each partition. Lower panel: When using the GTR20 model (i.e., the general-time-reversible model extended to the 20x20 aa substitution rate matrix), the 190 rate parameters can be optimized during the search (i.e., if the RPM operator is selected), but the starting values can be set to the values of any of the empirical models (WAG, JTT, ...) by selecting the model in the drop-down menu (red arrow 1), and hitting the 'Fill R matrix' button (red arrow 2).

**Note:** for nucleotide substitution models, the ‘transition-transversion ratio’ ( $Ti/Tv$ ) is the parameter called  $kappa$ , *i.e.*, the ratio between the rate of  $Ti$  and the rate of  $Tv$ . Because there are twice as many possible transversions ( $A \leftrightarrow T$ ;  $A \leftrightarrow C$ ;  $G \leftrightarrow T$ ;  $G \leftrightarrow C$ ) as possible transitions ( $A \leftrightarrow G$ ;  $T \leftrightarrow C$ ), the  $kappa$  parameter does not equate to the ratio ‘frequency of  $Ti$  / frequency of  $Tv$ ’. For example, under the JC model,  $kappa=1$  but ‘ $FreqTi/freqTv$ ’ = 0.5. For the codon substitution models  $kappa = 'freqTi/freqTv'$ .

**Note:** Model parameter values can be estimated from the NJ tree using the ‘Estimate starting parameters’ button (blue frame, Fig. 15). However, if you stop the estimation before it completes, parameter values will not be re-set to the original values but to the values obtained by the optimization algorithm right before it was stopped.

### Automated choice of best Model (LRT, AIC, BIC)

One difficulty in ML phylogeny inference is to choose the “right” substitution model: too-simple a model will fit the data poorly and can lead to erroneous inference, whereas too-complex a model will run more slowly and over-fit the data (*i.e.*, too many parameters in relation to the data will generate an increased variance for all parameters ... the model will describe noise in addition to the data). The softwares MODELTEST and PROTTEST (<http://darwin.uvigo.es>) implement statistical methods for selecting the model that best fits the data ([41]; and refs therein). MetaPIGA makes the procedure easier as it implements the **Likelihood Ratio Test**, the **Akaike Information Criterion**, and the **Bayesian Information Criterion** and performs parameter optimization automatically: simply choose your preferred model testing method (red frames in Fig. 15). For example, running the Akaike Information Criterion test on the ‘*ranoidea\_1b.nex*’ file will generate the results shown in Figure 16: MetaPIGA proposes to use the GTR model with gamma-rate heterogeneity but no proportion of invariant sites. Accepting this proposition will set this model in MetaPIGA as well as the starting parameter values (here, rate parameters and gamma distribution shape parameter) to those evaluated during the test. As the various models are tested in parallel on all the CPU cores of your machine, MetaPIGA will warn you if not enough memory is available, a problem that can easily be alleviated by reducing the number of cores allocated to the task (blue oval in Fig. 16).

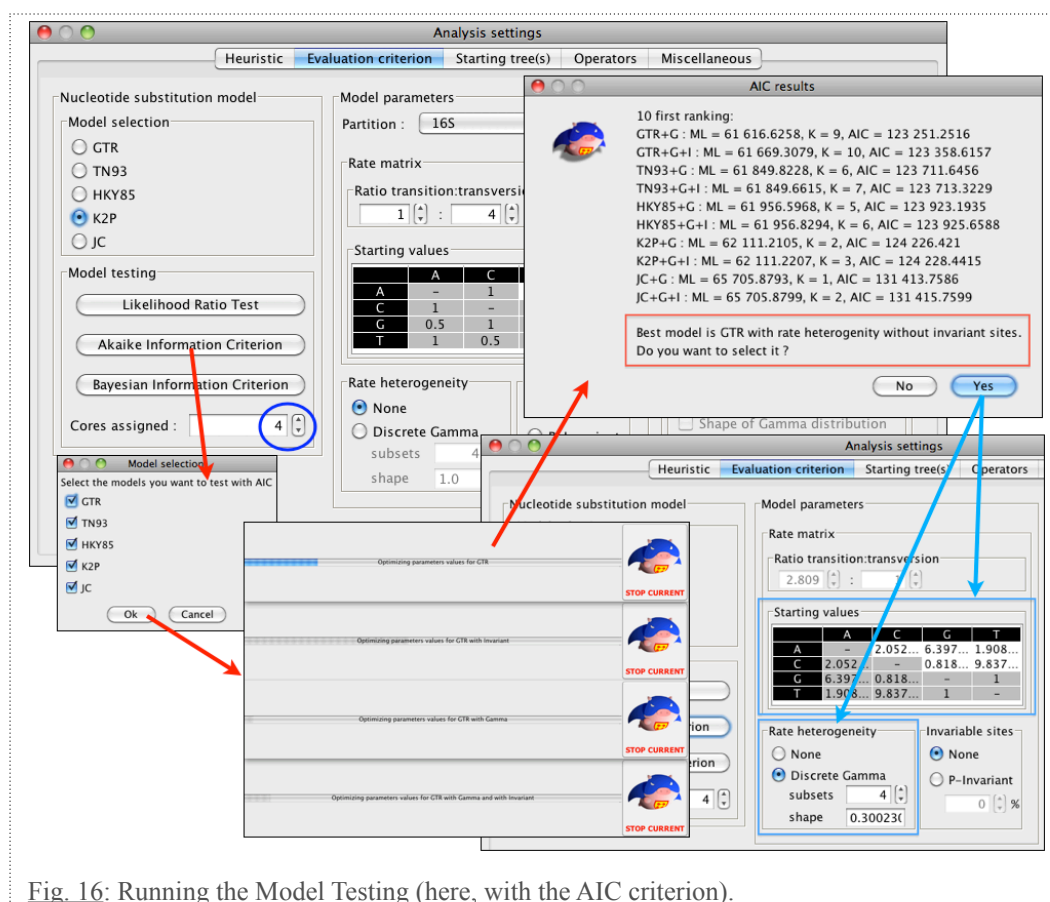


Fig. 16: Running the Model Testing (here, with the AIC criterion).



**Note:** that partitions (defined in the ‘Dataset Settings’ window, Figs. 4-10) are taken into account when performing a ‘Model Test’. Given that model testing can take several hours to run on large datasets (especially with protein data, given the number of models to compare), MetaPIGA allows you to restrict model testing (Fig. 16) to the comparison of a subset of models.

**Note:** if you want to abort model testing (*e.g.*, because you forgot to include/exclude taxa and/or charsets, or want to change your partitioning of the data), hit the ‘CANCEL TESTING’ button: testing will be aborted and all optimizations performed so far will be ignored. On the other hand, hitting a ‘CANCEL CURRENT’ button will stop optimization on the model being currently evaluated; obviously, the results of the statistical tests will then be contestable.

### Intra-step optimization

All parameters of the model (transition/transversion ratio or components of the rate matrix, the shape parameter of the  $\gamma$ -*distr*, and *Pinv*), branch lengths, and among-partition relative rates can experience ‘Intra-step optimization’ (blue frame in Fig. 15) either periodically during the search and/or at the end of the search. The principle of stochastic methods (*i.e.*, inter-step optimization methods), such as MC<sup>3</sup> approximations of the Bayesian approach, stochastic simulated annealing, and genetic algorithms, is to *AVOID* intra-step optimization. Hence, the default in MetaPIGA is that all target parameters (chosen by the user) are NOT optimized intra-step (only the consensus tree obtained after replicated searches -- see *section 5.4.5* -- will have its model parameters optimized). Hence, the stochastic heuristic itself will optimize topology, branch lengths and other model parameters during each search. When using the ‘discrete’ or ‘stochastic’ options (blue frame, Fig. 15), current best tree(s) are also optimized during the search, respectively every *s* numbers of steps or with a probability *p* at each step. These two options can obviously greatly increase running time.

**Note:** For intra-step optimization, MetaPIGA implements a single algorithm: a genetic algorithm without recombination; each tree to optimize is copied 7 times and the population of 8 individuals experiences mutations (on selected targets); selection is performed with tournament; the GA stops when the likelihood remains unchanged for 200 steps (generations). Future versions of MetaPIGA will also include alternatives to the GA (such as, possibly, the Powell’s algorithm).

**Note:** the ‘consensus tree only’ option (blue frame, Fig. 15) is equivalent to the “never” option when performing a single search (one replicate). The two options differ only when performing multiple replicates (see *section 5.4.5* below). When target parameters are optimized every *s* steps or stochastically, optimization is also performed at the end of (each) search.

### 5.4.3. The ‘Starting tree(s)’ tab

As shown in Figure 17, the user can choose to produce the starting tree(s) either as *NJ Tree(s)* [47] or as *Random Tree(s)* (*i.e.*, with random topology and random branch lengths) or as ‘*Loose Neighbor Joining*’ (*LNJ*) tree(s), *i.e.*, a pseudo-random topology (modified from [1]). For generating a LNJ tree, the user specifies a proportion value ( $p=[0-1]$ ) and, at each step of the NJ algorithm, the two nodes to cluster, instead of corresponding to the smallest distance value, are randomly chosen

from a list containing the  $\frac{NTax(Ntax-1)p}{2}$  smaller distances, where *NTax* is the number of se-

quences in the dataset. Branch lengths are computed as in the NJ method. **In other words, the LNJ tree is a NJ tree with some topology randomization which amount is defined by the user.** This approach is a particularly useful compromise between random starting trees ( $p=1$ ) that require long runs of the heuristic for optimization, and a good but fixed topology (the NJ tree, *i.e.*,  $p=0$ ) that might be prone to generate solutions around a local optimum. The LNJ starting tree method is particularly well adapted to the metaGA. Indeed, starting from  $I \times P$  (where *I* is the number of individuals (trees) per population and *P* is the number of populations) random trees will significantly increase the search time whereas starting from  $I \times P$  identical NJ trees will cause the stopping rule to be

reached too fast (see below) with local optima solutions. On the other hand, **LNJ starting trees provide enough variation among populations for avoiding local optima but significantly speed-up the search in comparison with using ‘True random’ starting trees.**

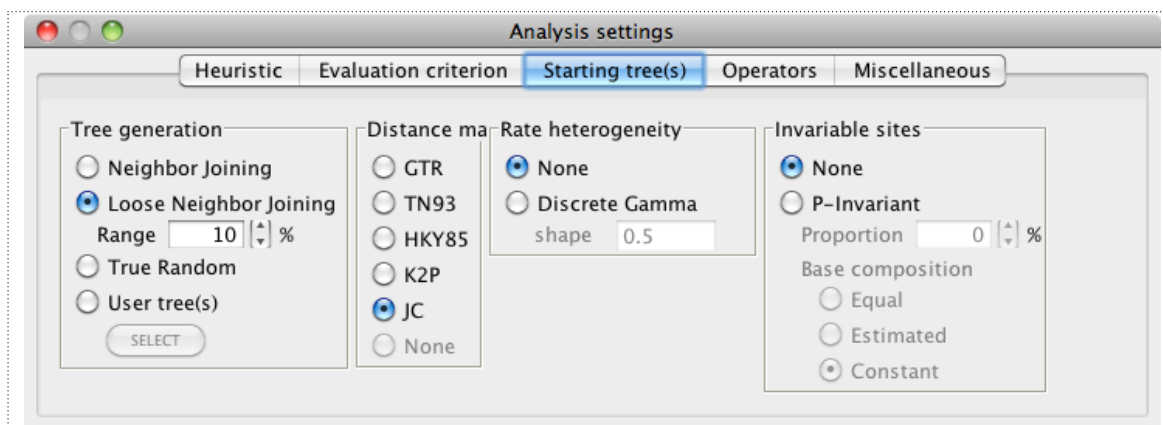


Fig. 17: The ‘Starting tree(s)’ window.

**Note:** The distance matrix used for building NJ or LNJ starting trees can be computed using any of the available substitution models (see above) and with or without *Pinv* and/or  $\gamma$ -*distr*. Unless the user wants to start with trees with the highest likelihood possible, we recommend using a simple and fast model (e.g., JC and Poisson respectively for nucleotide and protein data) for generating starting trees as they will anyway be highly modified during the heuristic search. For codon substitution models, three distance matrices are calculated (for codon positions 1, 2, and 3) using one of the available nucleotide substitution models. These three matrices are then weighted based on the evolutionary information they provide and combined into the single distance matrix [48].

**Note:** When choosing the ‘Neighbor Joining’ starting-tree option during a ‘Random-restart hill climbing’ search (Heuristic tab, section 5.4.1), the NJ tree will only be used for the first hill climbing, and ‘LNJ trees’ will be generated for all restarts.

**Note:** Arbitrary starting trees (in Newick format) can also be imported by the user. When clicking on the ‘User tree(s)’ radio button then on the ‘select’ button (Fig.17), you will be prompted to choose starting trees from a list. Various buttons allow you to add more trees in that list either from the ‘TreeViewer’ or from Nexus files.

**Notes:** if the Nexus file contains user trees (in a Tree Block) and if you select the ‘User tree(s)’ starting-tree option:

- ✓ The first tree in the Tree Block will be used if you selected SA or stochastic HC as the heuristic;
- ✓ The *I* first trees in the Tree Block will be used when selecting GA as the heuristic option with *I* individuals (one tree per individual);
- ✓ The *P* first trees in the Tree block will be used when selecting CP as the heuristic with *P* populations (one tree per population);
- ✓ If there are too few trees in the list of starting trees, MetaPIGA will cycle among the available trees;
- ✓ In the case of a ‘Random-restart hill climbing’ search (‘Heuristic’ tab, section 5.4.1), if the number of provided starting trees is smaller than  $N+1$  (i.e., the number of restarts plus 1), LNJ trees will be generated for the missing starting trees.

#### 5.4.4. The ‘Operators’ tab

All stochastic heuristics use *Operators*, i.e., the topology and parameters’ modifiers allowing the heuristic to explore solution space. In MetaPIGA, we implemented 5 operators for perturbing tree topology and 6 operators for perturbing model parameters (see below). These operators can be used in any combination, either at equal or user-defined frequencies. The user can choose for these frequencies to change dynamically during the search, i.e., MetaPIGA can periodically evaluate the relative gains in likelihood produced by each operator and adjust their frequencies proportionally<sup>5</sup>.

<sup>5</sup> If only some of the operators are made dynamic, their probabilities are assigned after subtracting from 1 the probabilities of the fixed operators. A minimum operator frequency can be set to prevent operators from being switched off. Indeed, an operator which is very inefficient early in the search could become efficient later in the search.



In the example given in figure 18, the evaluation of the operators' performances is computed every 100 generations, and the minimum frequency of any selected operator is set to 4%.

'Nearest Neighbor Interchange' (*NNI*), 'Subtree Pruning Regrafting' (*SPR*), and 'Tree Bisection Reconnection' (*TBR*) are classical branch-swapping algorithms used in many heuristics for phylogeny inference [21]. MetaPIGA also implements the following topology operators:

- ✓ 'Taxa Swap' (*TXS*):  $n$  randomly-selected terminal branches are randomly swapped. The value of  $n$  can be set to any number between 2 (default) and the total number of taxa (*ALL*), or randomly chosen (*RAND*) at each generation.
- ✓ 'Subtree Swap' (*STS*): 2 (default) or a random number (*RAND*) of subtrees are randomly swapped.

The 6 other operators affect model parameters:

- ✓ 'Branch Length Mutation' (*BLM*) and 'internal Branch length mutation' (*BLMint*). As our preliminary analyses (data not shown) indicated that branch length optimization yields external branch lengths that are quite similar to those obtained through topology-constrained NJ (both on a NJ topology and on a ML topology), we implemented a branch-length operator (*BLMint*) affecting internal branches only. We also implemented a branch-length operator (*BLM*) that can affect all (internal and external) branches.
- ✓ 'Rate Parameters Mutation' (*RPM*): This operator is not available for the *JC* model as the rate parameter is identical for all possible substitutions under this model. The *K2P* and *HKY* models consider two rates (the rate of  $T_i$ , and the rate of  $T_v$ ); hence, only the  $\kappa$  parameter (ratio of  $T_i$  and  $T_v$  rates) can be affected. The *TN93* model assigns 3 different rates: for transversions, for  $A \leftrightarrow G$  transitions, and for  $T \leftrightarrow C$  transitions. The *GTR* model allows assigning different rates for the 6 possible substitutions:  $A \leftrightarrow T$ ,  $A \leftrightarrow C$ ,  $G \leftrightarrow T$ ,  $G \leftrightarrow C$ ,  $A \leftrightarrow G$ , and  $T \leftrightarrow C$ . Under the *TN93* and *GTR* models, the user can choose that each *RPM* operation affects either '*I*' (default) randomly chosen rate parameter or '*ALL*' rate parameters. The '*I*' and '*ALL*' commands are equivalent under the *K2P* and *HKY* models because, although there are two rates, there is only one free rate parameter (the other one is set to 1).
- ✓ 'Gamma Distribution Mutation' (*GDM*): modifies the  $\gamma$ -*distr* shape parameter.
- ✓ 'Proportion of Invariable sites Mutation' (*PIM*): affects  $P_{inv}$ .
- ✓ 'Among-Partitions Rate Mutation' (*APRM*): affects the relative rates among partitions.

#### Notes:

- The *BLM*, *BLMint*, *RPM*, and *GDM* operators affect their corresponding parameter by multiplying the parameter's value of the previous generation by a random number drawn from an exponential distribution (with  $\lambda=2$ ), and shifted by 0.5 (such that the minimum value is 0.5 and the mean is 1).
- The *PIM* (values between 0 and 1) and *APRM* operators affect their corresponding parameters by multiplying the parameter's value of the previous generation by a random number drawn from a normal distribution (with mean=1 and SD= 0.5). The resulting multiplier is rejected if  $\leq 0.4$ .
- For LNJ starting trees, the initial length of all internal branches is computed with the NJ algorithm whereas,

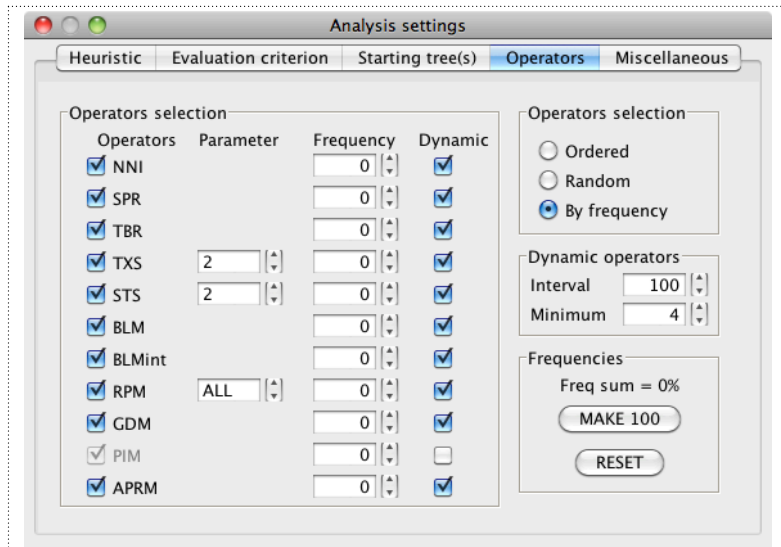


Fig. 18: The 'Operators' tab.



for random starting trees, they are drawn from an exponential distribution (with  $\lambda=1$ ), and shifted by 0.001 (to avoid zero length branches).

#### 5.4.5. The ‘Miscellaneous’ tab

This window allows the user to choose stop criteria and define the parameters of replicated searches (to obtain estimates of branches’ posterior probabilities). In addition, the user can choose which log files to save on disk. Also, if a supported graphics card is available, the user can choose to use either the CPU or the GPU for likelihood computation. Increase of computation speed is particularly significant for protein and codon models.

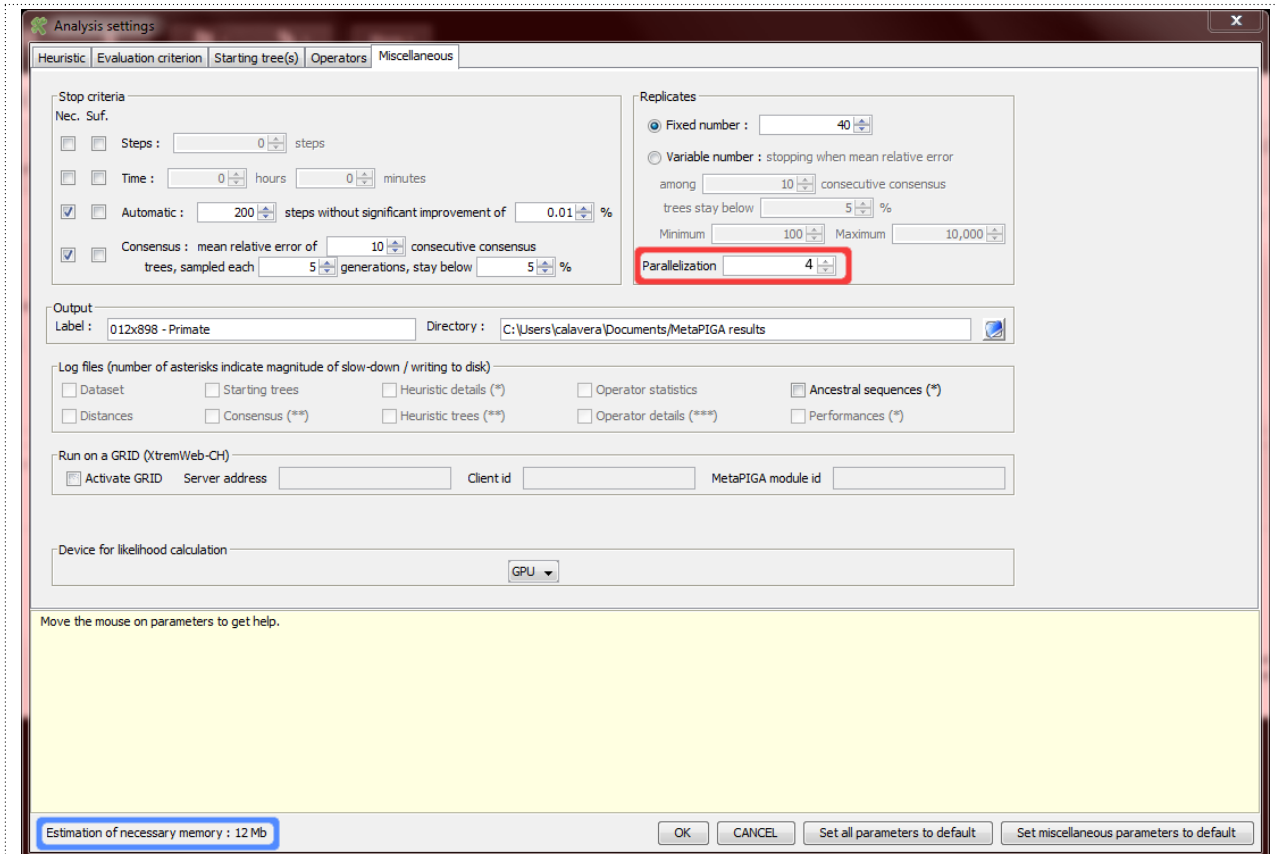


Fig. 19: The ‘Miscellaneous’ window for defining stopping condition(s), parameters for performing replicates (and obtaining estimates of posterior probabilities under the MetaGA), and the label of the directory in which all results will be saved. Log files to be saved on disk can also be defined. The amount of memory required for running the analysis (blue frame) has significantly increased because a complex model is used (Fig. 14) and because 4 cores have been chosen for parallelization (red frame).

#### Stop Criteria

Exactly as in the MC<sup>3</sup> approximations of the Bayesian approach [24, 25] implemented in the software MrBayes [31] for which the user must define a number of generations and trees to sample before stopping the search, all stochastic heuristics implemented in MetaPIGA require a stop condition. We implemented several stop conditions in MetaPIGA; any number of conditions can be set and each one can be necessary or sufficient (Fig. 19, ‘nec.’ or ‘suf.’)<sup>6</sup>. The stop criteria are: number of steps e.g., number of generations for the GA or the metaGA), elapsed time, and likelihood stability. The later, termed ‘Automatic’ in the GUI (Fig. 19), means that the search stops when the log-likelihood of the best tree has not improved of more than a given percentage (defined by the user,

<sup>6</sup> The heuristic stops when any of the sufficient conditions is met or when all necessary conditions are met. Conditions are sufficient by default.

0.05% by default) at any step during  $n$  steps ( $n$  also defined by the user).

**Note:** that, when using the ‘*Random-restart hill climbing*’ heuristic (Fig. 11), the stop conditions are defined for one hill climbing. For example, when using random-restart hill climbing with 10 restarts and ‘2000 steps’ as the stop condition, 11 hill climbing of 2000 steps will be performed but only the best scored tree, among the 11 results, will be kept.

When using the metaGA heuristic, one can use the **Consensus**’ stopping condition based on convergence of the populations of solutions. Indeed, comparing (across generations) the frequencies of internal branches shared among the  $P*I$  trees provides a means for assessing whether the populations converge towards a stable set of solutions, *i.e.*, towards a consensus with stable branch frequencies. Hence, a stopping rule, not available to other heuristics, can be used under Consensus Pruning (=MetaGA): the user can choose to stop the search when a series of mean relative error (MRE) values remains, across generations, below a threshold (in %) defined by the user. In our experience, using the Consensus stopping-rule with a threshold of 5% works very well when performing replicates (for estimating posterior probabilities of clades, see below). On the other hand, if you perform a single search in order to find the single very best tree, you might want to experiment with either lower threshold values (*e.g.*, 1%) or the stopping rule based on stability of the likelihood value (*e.g.*, 200 steps without improvement of 0.01% of the log-likelihood value).

**Note:** To increase independence among samples, consensus trees are sampled every  $n>1$  (*i.e.*, non-successive) generations. For example, given two consensus tree,  $T_i$  and  $T_j$ , corresponding to the consensus among the  $P*I$  trees at generations 5000 and 5005, respectively, the MRE is computed as follows:

$$MRE(T_i, T_j) = \frac{\sum_{p=1}^{nPartition} \left| \frac{\Phi_{T_i}^p - \Phi_{T_j}^p}{\max(\Phi_{T_i}^p, \Phi_{T_j}^p)} \right|}{nPartition}, \text{ where } nPartition \text{ is the sum of taxa bi-partitions observed in } T_i \text{ and } T_j$$

(but identical partitions are counted once), and  $\Phi_{T_i}^p$  and  $\Phi_{T_j}^p$  are the consensus values of bi-partition  $p$  in trees  $T_i$

and  $T_j$ , respectively. Note that  $\left| \frac{\Phi_{T_i}^p - \Phi_{T_j}^p}{\max(\Phi_{T_i}^p, \Phi_{T_j}^p)} \right| = 1$  if either both  $\Phi_{T_i}^p$  and  $\Phi_{T_j}^p$  are nil, or if the corresponding

internal branch does not exist in either  $T_i$  or  $T_j$ . Internal branches that are absent from both  $T_i$  and  $T_j$  are not considered. If the  $MRE_{(gen5000, gen5005)}$  is above the user-defined threshold (*e.g.*, 3%), it is discarded and a new MRE is computed for the comparison of generations 5005 and 5010. On the other hand, if  $MRE_{(gen5000, gen5005)}$  is below the threshold, a counter is incremented and a new MRE is computed for the comparison of generations 5000 with the next sample (here, corresponding to generation 5010). The user defines for how many samples the MRE must remain below the specified threshold before the search stops.

## Replicates

**This functionality is very important because it allows estimating the support of trees and clades.** For all stochastic heuristics implemented in MetaPIGA, the user can chose to repeat the search many times, generating a majority-rule consensus tree among the replicates. This is particularly useful under the metaGA because previous analyses [1] indicate that **a set of multiple metaGA searches produces trees and clades with frequencies that approximate their posterior probabilities**. Hence, metaGA branch support values would be comparable to posterior probabilities provided by MC<sup>3</sup> approximations of Bayesian approaches. The user can either fix the number of replicates, or specify a range of minimum and maximum number of replicates then choose to let MetaPIGA stop automatically, exploiting the MRE metric in a similar way as the consensus across populations in a single metaGA search (see above).

**Note:** Here, however, the MRE is computed using consensus across replicates, *i.e.*,  $T_i$  is the consensus among the final trees obtained between replicates 1 and  $i$ . No additional replicate is produced when the MRE among  $N$  replicates remains below a given threshold. Consecutive replicates can be used because they are independent. As an example, if  $N$  is set to 10, and the first MRE below the user-defined threshold (*e.g.*, 5%) in-

volves replicates 1-241 and 1-242, the MRE is computed 9 additional times, *i.e.*, between the reference consensus  $T_{1-241}$  and  $T_j$ , for  $j$  corresponding to replicates 1-243, then 1-244, then 1-245, etc. The search stops if the inter-replicates MRE remains below 5% for 10 consecutive replicates. On the other hand, the counter is reset to zero as soon as the MRE exceeds 5%, and the new reference tree for computing the MRE is then set to  $T_{1-current\ replicate}$ . The inter-generations (=intra-replicate) MRE stopping rule can be used in combination with the inter-replicate MRE stopping rule, letting MetaPIGA decide both when to stop each replicate and when to stop executing additional replicates (*i.e.*, when to stop the entire analysis).

**Note:** in most cases, performing multiple replicates is aimed at generating a consensus tree and estimating support of internal branches, hence, it is usually not important to perform a final intra-step optimization of all model parameters at the end of each replicate. This is why the default for 'Intra-step optimization' (blue frames, Fig. 15) is '**consensus tree only**'. It means that a final round of optimization for branch lengths and model parameters is NOT performed after each replicate (this will significantly save run time and will not change anything to the internal branches' frequencies) but it is performed on the final consensus tree (*i.e.*, model parameters and branch lengths are optimized on the consensus-tree topology). When the user chooses to optimize best trees '*at the end of (each) search*' the consensus tree is optimized as well. With the '*discrete*' and '*stochastic*' options, current best tree(s) are also optimized multiple times during each replicate as well as at the end of each search.

## The grid

To start an analysis on an XtremWeb-CH Grid, check the 'Activate GRID' check-box and write your grid credentials in the appropriate boxes. If your account is active and the MetaPIGA binaries are uploaded to the MetaPIGA module, your analysis will start on the Grid after you press the 'Run' button. For user documentation, please refer to the following site: [www.xtremwebch.net/mediawiki/index.php/How\\_use](http://www.xtremwebch.net/mediawiki/index.php/How_use). Please, contact us for additional information.

## Log files

The user can choose to write log files on disk. This is however mostly for debugging purposes and performance testing such that only expert users might need this functionality. Selecting the log files indicated with asterisks can (i) significantly slow down the search and (ii) fill up large amount of disk space (with the magnitude of slow-down and fill-up approximately indicated by the number of asterisks). All log files are written in the results folder (see below).

- ✓ Dataset - Working matrix log file - Prints the compressed dataset into 'Dataset.log'. The last row contains the weight of each column, *i.e.*, the number of times this data pattern is found in the data matrix.
- ✓ Distances - Distance matrix log file - Prints the distance matrix into 'Distances.log'.
- ✓ Starting trees - Starting Trees log file - Prints the starting tree(s) into 'StartingTrees.tre'.
- ✓ Consensus (\*\*) - Consensus log file - The 'Consensus.log' file records consensus at each step of Consensus Pruning. It requires disk space between 100 bytes and 1Kb per taxa and per consensus recorded. For example, recording consensus for a dataset of 200 taxa, using the metaGA heuristic for a fixed number of 5000 generations will generate a file between 100Mb and 1Gb for each replicate produced.
- ✓ Heuristic details (\*) - Heuristic search log file - The 'Heuristic.log' file records details about each step of the heuristic. Requires disk space between 500 bytes & 1 Kb per iteration of the heuristic.
- ✓ Heuristic trees (\*\*) - Heuristic search tree file - The 'Heuristic.tre' file records each tree found at each step of the heuristic. It requires disk space of +/- 130 bytes per taxa per tree recorded. For example, recording trees for a dataset of 200 taxa, using the metaGA heuristic with 4 populations of 4 individuals each, for a fixed amount of 5000 generations will generate a file of about 1.5Gb for each replicate produced.
- ✓ Operator statistics - Operator statistics file - The 'OperatorsStatistics.log' file records operator statistics at the end of a search, as well as each time the operator frequencies have been updated.
- ✓ Operator details (\*\*\*) - Operators log file - The 'OperatorsDetails.log' file records details about the operators used. It requires disk space of 200-300 bytes per taxa per operation. For example, recording operator details for a dataset of 200 taxa, using the metaGA heuristic with 4 populations of 4 individuals each, for a fixed number of 5000 generations will generate a file between 1.7Gb and 3.4Gb for each replicate produced.
- ✓ Ancestral sequences - Ancestral sequences log file - At the end of the heuristic, the ancestral sequence probabilities are printed into the 'AncestralSequences.log' file



- ✓ **Performances (\*)** – The ‘*Performances.log*’ file records the amount of time (in nanoseconds) used by each operator. It requires disk space of +/- 1 Kb per iteration of the heuristic.

### Output label and directory

Viewing of the analyses results can be done in the MetaPIGA graphical interface. However, all results are also written on disk for later retrieval, viewing and manipulation. When a MetaPIGA search is started, a result directory (named ‘*MetaPIGA results*’) is generated in your home directory (Mac OS X & Linux) or in the ‘*My documents*’ folder (Windows). When you launch an analysis, the results will be automatically saved in a folder named with its ‘**label**’ (which is, by default, the name of the nexus file minus the “*nex*” extension, see Fig. 19) followed by the date (year-month-day) followed by the time (hour\_min\_sec) at which the search was started. This allows for easy differentiation of analyses performed at different times on the same dataset. For example, the result folder “*ranoidea\_1b - 2010-06-09 - 17\_16\_16*” includes the result files for the analysis of the “*ranoidea\_1b.nex*” data set started on June 9, 2010 at 5:16:16PM.

At the end of the search, the result folder contains the file ***Results.nex***, *i.e.*, a text file including:

- ➡ A MetaPIGA block corresponding to the search parameters;
- ➡ The data set;
- ➡ A tree block with the result trees, *i.e.*:
  - \* Either (if replicates have not been performed):
    - The best tree found (among all *P\*I* trees) named appropriately (*e.g.*, ‘*TREE rana\_~\_2010~06~09\_~\_17\_16\_16\_~\_Genetic\_algorithm\_best\_solution*’)
    - The best tree, of each of the *P* populations, named appropriately (*e.g.*, ‘*TREE rana\_~\_2010~06~09\_~\_17\_16\_16\_~\_Best\_individual\_of\_population\_0*’)
  - \* Or (if replicates have been performed):
    - The consensus tree (among all replicates) named appropriately (*e.g.*, ‘*TREE rana\_~\_2010~06~09\_~\_17\_16\_16\_~\_Consensus\_tree\_~\_200\_replicates*’)
    - Then, for each replicate:
      - The best tree found (among all *P\*I* trees) named appropriately (*e.g.*, ‘*TREE rana\_~\_2010~06~09\_~\_17\_16\_16\_~\_Genetic\_algorithm\_best\_solution\_[Rep\_8]*’)
      - The best tree, of each of the *P* populations, named appropriately (*e.g.*, ‘*TREE rana\_~\_2010~06~09\_~\_17\_16\_16\_~\_Best\_individual\_of\_population\_0\_[Rep\_8]*’)

If replicates have been performed, the result folder will also contain a text file ‘***ConsensusTree.tre***’ with the consensus tree among replicates. That tree is automatically updated in the run directory after each replicate. **Hence, if a crash or power cut occurs, the latest consensus tree (summarizing all replicates that accumulated before the cut) can be loaded and visualized in MetaPIGA after restarting.** As the name of the tree includes the number of replicates, you will know when the cut occurred.

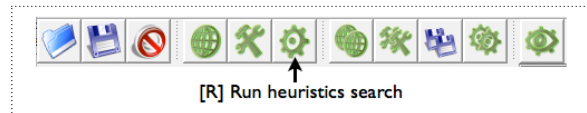
As the consensus tree file is in Newick format, it can also be loaded in tree viewing softwares such as *FigTree* (<http://tree.bio.ed.ac.uk/software/figtree/>) or *TreeView* (<http://taxonomy.zoology.gla.ac.uk/rod/treeview.html>).

If log files have been requested (see above), they will be printed either in the results folder or in corresponding replicates subfolders.


### 5.4.6. Exiting the Settings Window

Once all settings have been chosen by the user for the ‘*Heuristic*’, ‘*Evaluation Criterion*’, ‘*Starting Tree(s)*’, ‘*Operators*’, and ‘*Miscellaneous*’ tabs and the OK button has been hit, the Settings window closes and the main (entry) window is updated with the new settings listed in the upper-right window. The user can go back to the setting window at anytime for changing any parameter. Switching to another dataset in the left window and modifying the settings for that dataset does not affect the settings associates to the other datasets.

## 5.5. [R] The Run window



The search is launched by clicking on the

**‘[R] Run heuristic search’** button  (or by selecting in the menu: ‘Search’ → ‘Run’). Once the starting trees have been generated (this can take time), the user can follow the ongoing search by looking at the lower left panel of the run window which displays graphical information specific to the chosen heuristic method. For example, figure 20 shows the running window for a MetaGA search with replicates, 4 populations, and stopping rules as indicated in figure 19. The lower-left panel indicates the likelihood progression of each of the populations (the best tree likelihood in each population is indicated) as well as which replicate is ongoing (rep 71). If you set replicates parallelization to >1 (see red frame in Fig. 19), tabs give access to the graphs corresponding to each CPU core (core number 2 is selected in Fig. 20; red arrow).

When using the Stochastic Hill Climbing (HC) or the simple Genetic Algorithm (GA), the lower-left panel displays the likelihood progression of either the current tree (for stochastic HC) or of the best tree in the single population of trees (for the GA). When using the SSA, it indicates the progression of both the *‘temperature’* and of the likelihood. During a random-restart Hill Climbing, the graphical interface indicates the likelihood of the overall best solution (green line), the best solution of the current restart (yellow curve), and the starting tree of each restart (red line). Magenta and blue vertical lines indicate new restarts and replicates, respectively.

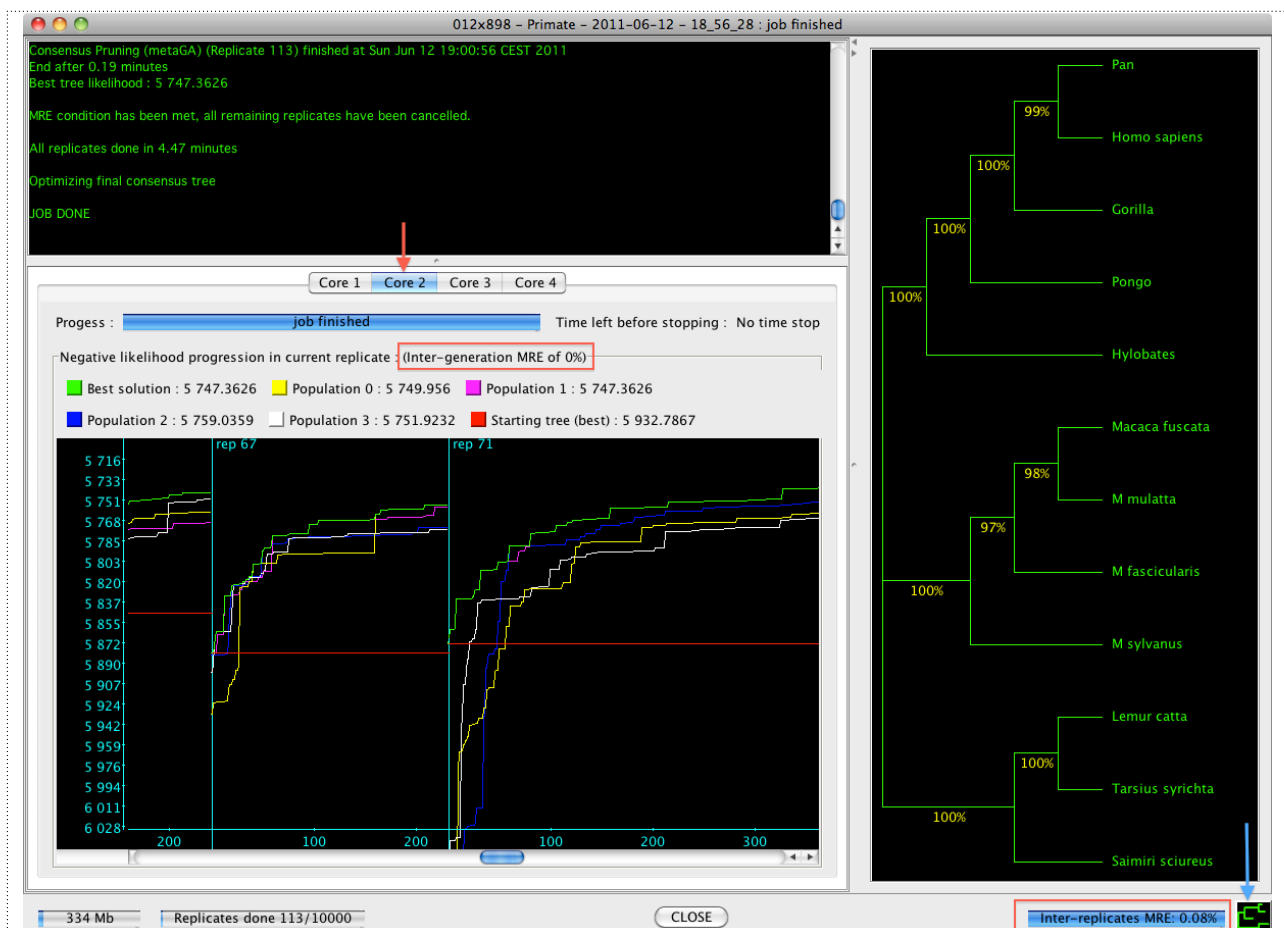


Fig. 20: The run window when replicates have been requested under the MetaGA heuristic. The best tree likelihood in each population is indicated



In parallel with the likelihood progression displayed in the lower-left panel, the right panel displays information on the current inferred phylogeny. When performing a single search (*i.e.*, without replicates) the tree displayed is the current best tree. When performing replicates (as in Fig. 20), the right panel shows the current consensus tree (and corresponding frequencies of internal branches) among all replicates accumulated thus far. Hence, the right panel of the run window allows the user to observe, on the fly, the progression of the phylogeny inference or (when using the metaGA) the progression of posterior probabilities of branches. In both cases, the user can switch between phylogram and cladogram (blue arrow in Fig. 20). The current values of the inter-generations (=intra-replicate) MRE and inter-replicate MRE (see ‘stopping rules’ above) are also indicated (red frames in Fig. 20).

Once the search is completed, a window will pop up reminding you that all results (best trees and consensus tree) have been saved in your result folder, but will also propose you to send ‘*All best trees*’ or the ‘*Consensus tree only*’ to the ‘*Tree Viewer*’ (see section 5.6 below) for further manipulations (rerooting, exporting, changing substitution model and further optimizing model parameters, reconstruction of ancestral states, etc).

When using the **XtremWeb-CH Grid**, the run window shows the status of the workers: queued, waiting, processing, completed, killed, or in error (see Fig. 21 for details). Workers with the status ‘*complete*’ have already sent their results back to your local machine. To use the XtremWeb Grid, please, refer to the following web site:

[http://www.xtremwebch.net/mediawiki/index.php/How\\_use](http://www.xtremwebch.net/mediawiki/index.php/How_use)

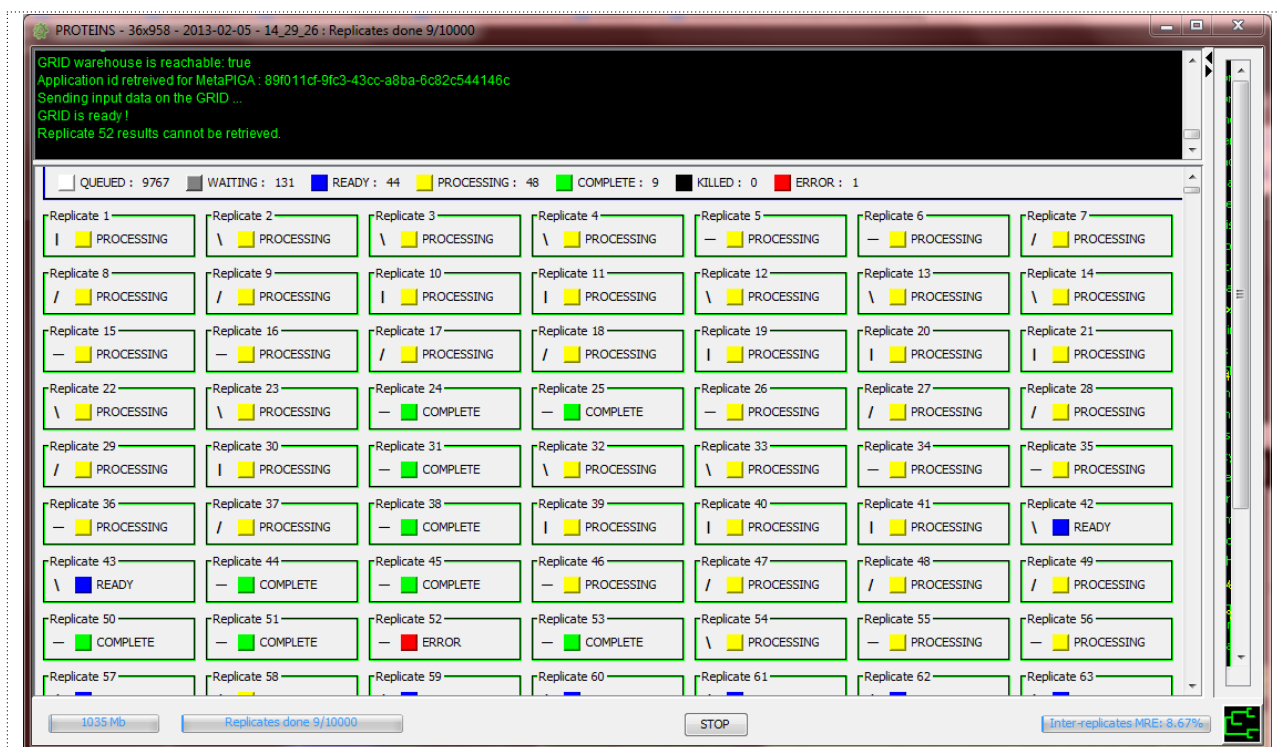


Fig. 21: The ‘Run’ window when using a XWCH grid. Status of worker are color-coded. A white box indicates that the replicate is waiting to be submitted to the grid whereas a gray box means that the replicate is waiting for an available worker. A blue box indicates that the worker is selected will start the analysis. A yellow box indicates that the replicate is running. A green box indicates that the replicate is completed and successfully retrieved from the grid. Finally, a red box means that MetaPIGA cannot retrieve the result or that the worker is not responding (error replicates are not used and have no effect on your analysis). Replicates stopped by the user are indicated with a black box (‘*killed*’).




## 5.6. [T] The tree viewer



### 5.6.1. Viewing and evaluating trees

The Tree Viewer is opened by clicking on the

'[T] Tree viewer' button  (or by selecting in the menu: 'Tools' → 'Tree viewer'). Trees can be saved in the 'Tree viewer' either at the end of a search or by importing trees from files. The user can even type (or copy-paste) a tree in Newick format in the lower left panel (red frame in Fig. 22), give it a name and add it to the TreeViewer's list of trees. The Tree viewer allows to display, rename, or remove any of these trees at any time. The 'Clear list' button delete all trees from the Tree viewer. Buttons at the bottom of the right panel allow to display the selected tree in various styles (rectangular, triangular, circular, phylogram), and show/hide its nodes' numbers or its branch lengths. Other buttons allow rerooting a tree at any node, and save or print one or several selected tree(s). The upper right panel indicates the parameters of the model (for each partition, if any) and the corresponding likelihood (yellow oval; Fig.22) of the selected tree. Obviously, for computing a likelihood, every tree must be associated to a dataset, hence, the 'Tree Viewer' only lists the trees that are relevant to the active dataset. The latter can be selected either in the 'Current dataset' scroll down list (Fig. 23) or in the MetaPIGA main window (Fig. 2 and 10). This allows the user to easily manage trees generated with different datasets.

**Note:** Several trees can be simultaneously selected from the list by using 'command click' and/or 'shift-click'. This allows removing several trees simultaneously. On the other hand, all other commands (model change, printing, rooting, ancestral state reconstruction, etc.) will affect only the tree highest in the list of selected trees.

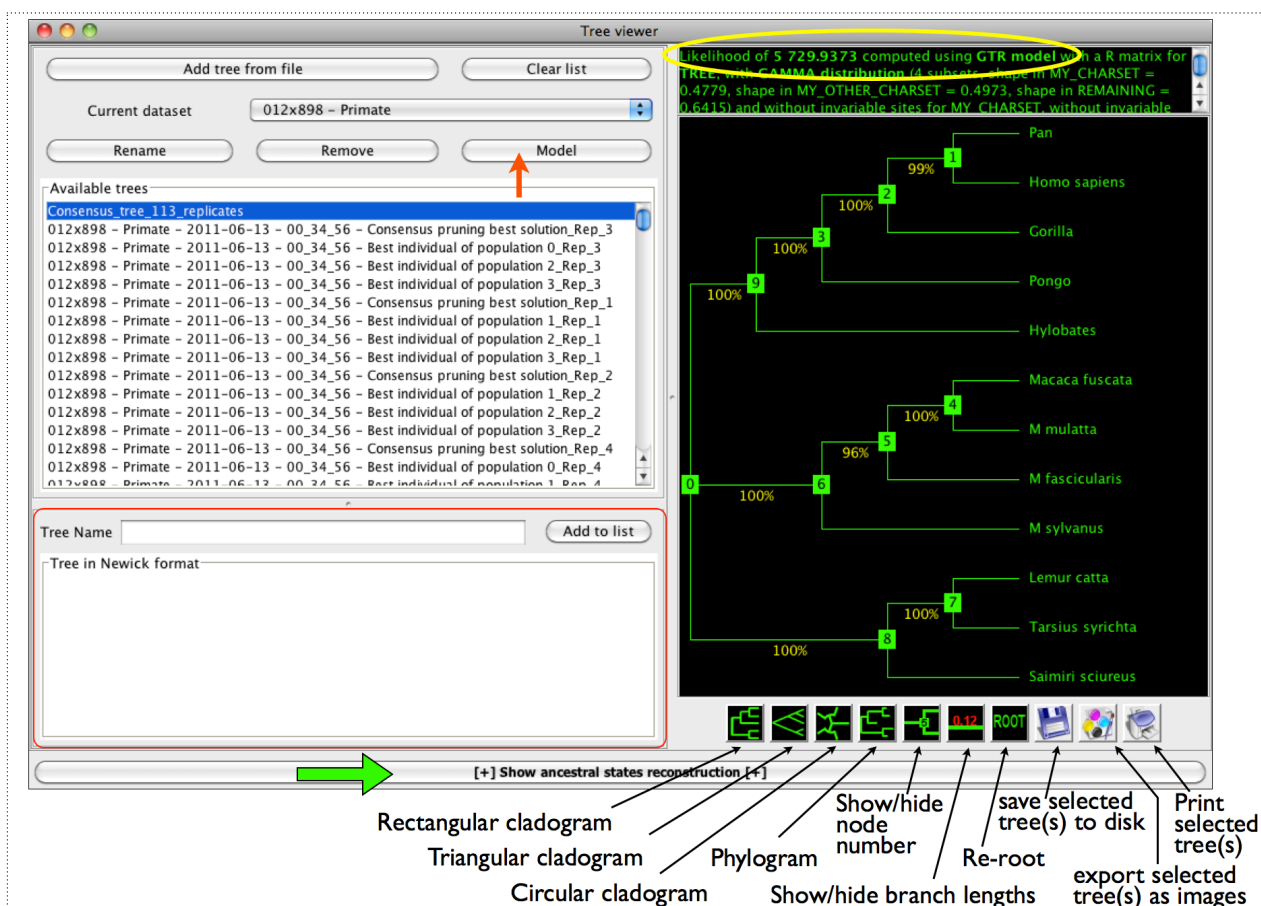


Fig. 22: The MetaPIGA tree viewer with one tree selected in the list. Red arrow: the 'Model' button gives access to a window (Fig. 23) for optimization of parameters and/or branch lengths and re-computation of the corresponding Likelihood under any substitution model. Green arrow: button giving access to the ancestral state reconstruction panel (Fig. 24).



Clicking on the ‘*Model*’ button in the ‘*TreeViewer*’ (red arrow; Fig. 22), opens the ‘*Evaluation settings*’ window (Fig. 23) that allows (for the selected tree) to (re-)optimize model parameters and/or branch lengths and re-compute the corresponding likelihood under any settings of model parameters. Note that, although parameters can be manually set for each partition separately by using the vertical tabs (Fig. 23), clicking on the ‘*Optimize model parameters*’ or ‘*Optimize branch lengths*’ button will perform joint optimization for all partitions. Once the model setting have been confirmed by clicking the ‘*OK*’ button, the upper-right panel of the ‘*TreeViewer*’ (Fig. 22), and the tree itself, will be updated with the new parameter values.

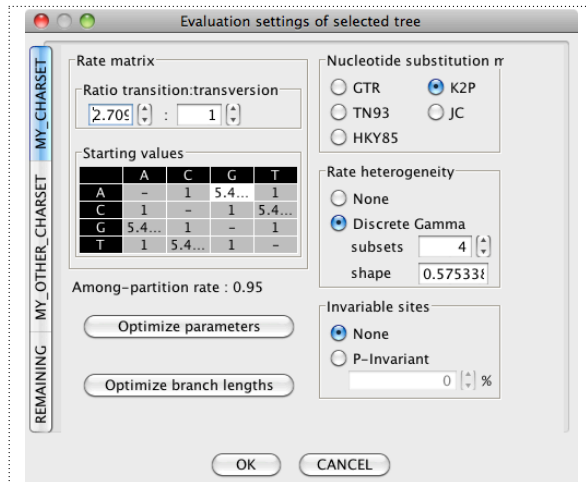


Fig. 23: The evaluation settings window.

### 5.6.2. Ancestral states reconstruction

During phylogeny inference under ML, the probabilities of all possible character states at all nodes are computed for all characters. This provides means for reconstructing ancestral sequences both *in silico* and in the laboratory (e.g., [10-14]). Clicking on the button indicated with a green arrow in Fig. 22 gives access to the ancestral state reconstruction panel of the ‘*TreeViewer*’. Simply select an internal node on the tree for viewing its corresponding ancestral sequence. Various buttons allow for different display styles and for exporting the ancestral sequence(s) (and the corresponding statistics) either of the selected node or of all internal nodes of the tree. The ancestral sequence reconstruction we implemented is Empirical Bayes [49].

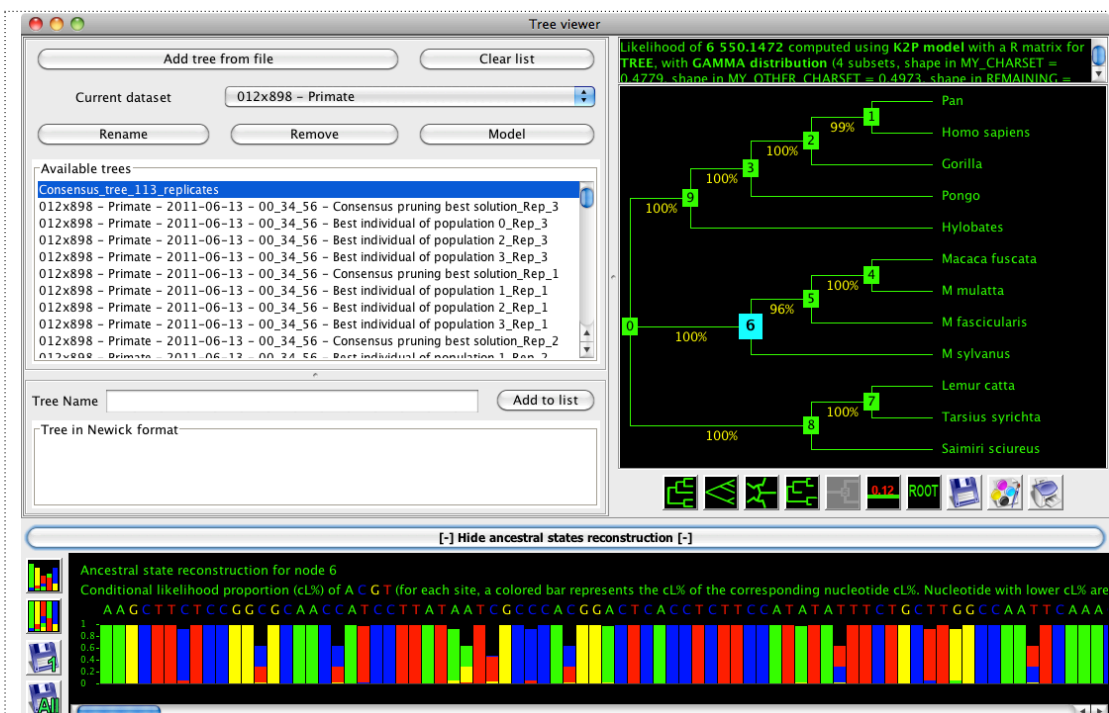




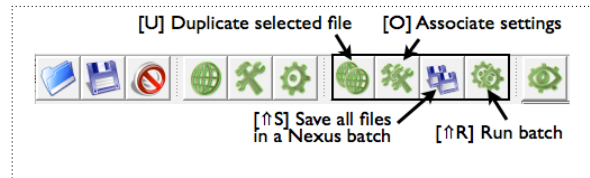


Fig. 24: The ancestral state reconstruction panel displays the conditional likelihood proportions of each state at each site for the node 6, directly selected on the tree in the upper-right panel. The  and  buttons allow exporting to disk a text file with the ancestral states of the selected node or of all nodes, respectively. Use the  and  buttons to switch between a view where bars of the histogram, for each character, are in front of each others (with the column of lowest likelihood proportion in the front) and a stacked histogram. The sequence indicated at the top corresponds to the most likely ancestral sequence.

## 5.7. Building and running batch files with the GUI

MetaPIGA supports the use of batch files that can be either written manually (see next *Section*) or generated using tools available in the GUI: datasets and their settings can be duplicated, settings can be “stamped” from one dataset to another, and multiple combinations of datasets and settings can be saved in a batch file that can be run either in the GUI (with various graphical information on search progress) or using command line.



### 5.7.1. Transferring analysis settings among datasets

Batch files are particularly useful for running different datasets with the same analysis settings. Imagine for example that you have opened 4 datasets in MetaPIGA ('012x898 - Primate', 'PROTEINS - 36x958', 'ranoidea\_1b', and '055x1314 - mp1') and that you have chosen all settings (using the various tabs in the *Analysis Settings* window, see *section 5.4*) for the dataset *ranoidea\_1b*. Now, as shown in figure 25, you can transfer these settings to any combination of other opened datasets by (i) choosing the source dataset, then (ii) clicking on the '[O] Associate settings' button (or by selecting in the menu: 'Batch' → 'Associate selected dataset analysis settings'), then (iii) selecting the dataset(s) you want to transfer the settings to, and (iv) click on the 'Associate' button.

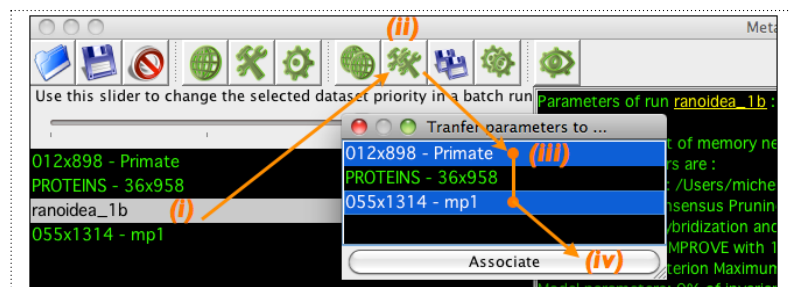


Fig. 25: Transferring settings from one dataset to other dataset(s).

- ✓ The batch can then be run in the GUI by clicking on the '[↑R] Run batch' button (or by selecting in the menu: 'Batch' → 'Run all datasets in a batch')
- ✓ Alternatively, the user can save the batch by clicking on the 'Save all files in a Nexus batch' button (or by selecting in the menu: 'Batch' → 'Save all datasets in a batch Nexus file'). This file can be run in command line (e.g., on a distant server) or re-imported in MetaPIGA and run through the GUI.

### 5.7.2. Duplicating datasets for batch files

Batch files are equally useful for running sequentially a single data set under multiple different settings: for example analyzing your favorite dataset with different substitution models or with different heuristics. First make as many duplicates of your dataset (called '012x898 - Primate' in Fig. 26) as you wish by clicking on the '[U] Duplicate selected dataset' button (or by selecting in the menu: 'Batch' → 'Duplicate selected dataset'). Then, select a duplicate and change the settings as required (in the 'Analysis settings' window). In this way, you can for example run a batch file that will sequentially run the 'Primate', 'Primate\_1', 'Primate\_2', and 'Primate\_3' datasets with, respectively the JC, K2P, HKY, and GTR substitution models. Note that, when duplicating a file, the settings listed in the 'Dataset settings' window (outgroup taxa, charsets, partitions, etc.) are duplicated as well.

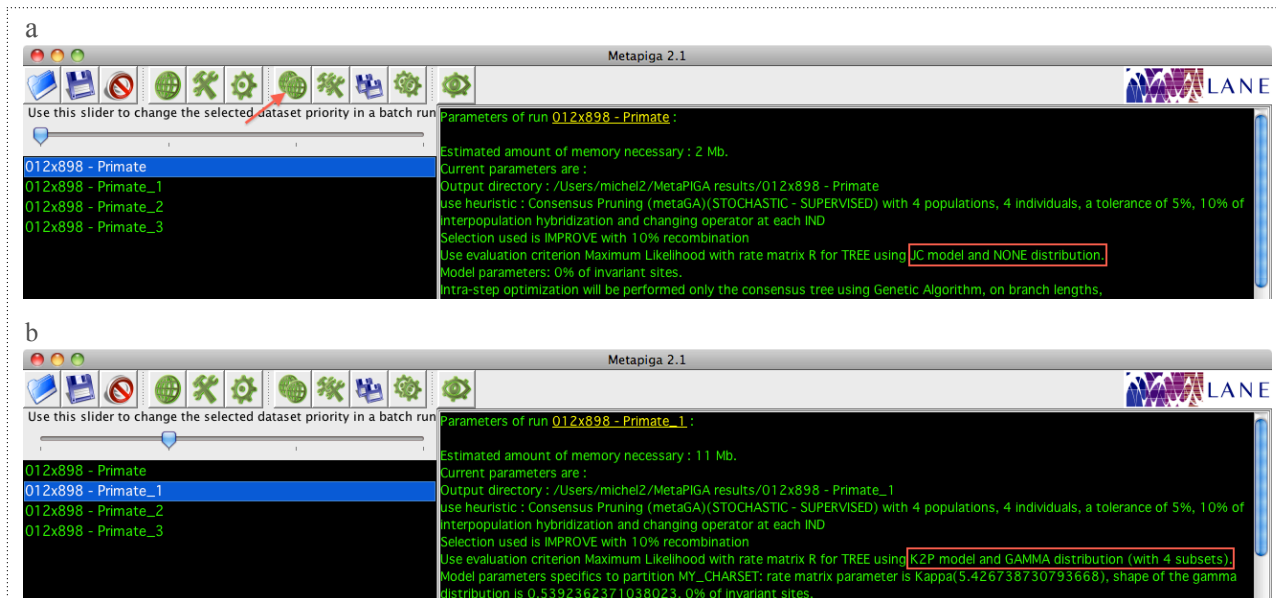


Fig. 26: (a) Duplicating datasets and (b) the parameter panel indicates the modified settings as chosen in the '[A] Analysis Settings' window (here, we have changed the substitution model to K2P with rate heterogeneity (and estimated starting parameters) for the 'Primate\_1' duplicate whereas the initial settings for the 'Primate' dataset is JC).

**Notes:** when running a batch file:

- ✓ The run window is simplified (in comparison to what is described above in *Section 5.5*). Beside basic statistics on the current run, the batch run window displays (in two separate panels) the log file information on the current run and on the overall batch. Two buttons allow for stopping either the current run or the entire batch of runs.
- ✓ The result trees of all replicates of all runs are automatically added to the 'TreeViewer'.

## 5.8. Building batch files manually

Instead of using the GUI, you can manually build Nexus batch files. As an example, the file below will run the single dataset of 15 taxa and 100 characters first with the JC model then with the GTR model + gamma-distributed rate heterogeneity.

The full list of MetaPIGA commands for manually building batch files are available in the *Appendix 1*.

**Check the end of section 5.2 for instructions on how running MetaPIGA in command line (this is particularly useful if you want to send jobs to a distant server).**

```
#NEXUS
[Metapiga - LANE (Laboratory of Artificial and Natural Evolution, University of Geneva)]

BEGIN BATCH;
RUN LABEL=15-100 DATA=data_1 PARAM=param_1;
RUN LABEL=15-100_1 DATA=data_1 PARAM=param_2;
END;

BEGIN METAPIGA;
[BATCHLABEL=param_1]
HEURISTIC CP CONSENSUS=STOCHASTIC OPERATOR=SUPERVISED NPOP=4 NIND=4 TOLERANCE=0.05
HYBRIDIZATION=0.1 SELECTION=IMPROVE RECOMBINATION=0.1 OPERATORAPPLIEDTO=IND NCORE=1;
EVALUATION MODEL=JC DISTRIBUTION=NONE PINV=0.0;
OPTIMIZATION ENDONLY ALGO=GA TARGET{ BL };
STARTINGTREE GENERATION=LNJ(0.1)
MODEL=JC DISTRIBUTION=NONE PINV=0.0;
```



```

OPERATORS { TXS(2) STS(2) TBR NNI SPR BLM } SELECTION=RANDOM;
SETTINGS LABEL=15-100;
STOPAFTER AUTO=200 CONSENSUS MRE=0.03 GENERATION=5 INTERVAL=10;
REPLICATES AUTOSTOP=MRE(0.05) RMIN=100 RMAX=10000 INTERVAL=10 PARALLEL=1;
OUTGROUP { ANABAENA_SP2 };
END;

BEGIN METAPIGA;
[BATCHLABEL=param_2]
HEURISTIC CP CONSENSUS=STOCHASTIC OPERATOR=SUPERVISED NPOP=4 NIND=4 TOLERANCE=0.05
HYBRIDIZATION=0.1 SELECTION=IMPROVE RECOMBINATION=0.1 OPERATORAPPLIEDTO=IND NCORE=1;
EVALUATION
MODEL=GTR RATEPARAM{ A(0.5) B(0.5) C(0.5) D(0.5) E(0.5) } DISTRIBUTION=GAMMA(4)
DISTSHAPE=1.0 PINV=0.0;
OPTIMIZATION ENDONLY ALGO=GA TARGET{ BL R GAMMA };
STARTINGTREE GENERATION=LNJ(0.1) MODEL=JC DISTRIBUTION=NONE PINV=0.0;
OPERATORS { TXS(2) STS(2) TBR NNI SPR BLM RPM(ALL) GDM } SELECTION=RANDOM;
SETTINGS LABEL=15-100_1;
STOPAFTER AUTO=200 CONSENSUS MRE=0.03 GENERATION=5 INTERVAL=10;
REPLICATES AUTOSTOP=MRE(0.05) RMIN=100 RMAX=10000 INTERVAL=10 PARALLEL=1;
OUTGROUP { ANABAENA_SP2 };
END;

BEGIN DATA;
[BATCHLABEL=data_1]
DIMENSIONS NTAX=15 NCHAR=100;
FORMAT DATATYPE=DNA MISSING=? GAP=- SYMBOLS="01" LABELS ITEMS=STATES STATESFORMAT=
=STATESPRESENT NOTOKENS;

MATRIX
Anabaena_sp2      CAAGATTACAGACTAAGTTATTACACACCTGATTACACACCTAAAGATACAGATATTTCTGGCGGCATTCCGTGTACACCCAGCCCGGAGTTCCTTTG
Chara_conniv      AAAGATTACAGATTAACTTACTATACCTGAGTATAAACTAAAGATACTGACATTTTAGCTGCATTTTCGTGTAACCTCCACACCTGGCGTTCACCTG
Chlor_ell         AAAGACTACCGTTTAACTTACTATACCTGATTACCAACCAAGACACTGATATTTTCGAGCGTTCGGTATGACTCCTCAACAGGTGTTCACCTG
Volvox_ro         AAAGATTATCGTTTAACTACTACACACCTGACTATGTAGTAAAGACACTGACATCTTAGCAGCATTTTCGTATGACTCCACAACAGGTGTTCACCTG
Sirogonium_melanosp AAAGATTACAGACTTACATATTACACTCCTGAATATGAGACCAAGAAAGAACTGATATTTTAGCTGCATTTCCGCATGACTCCTCAGCCTGGAGTACCACCTG
Zygnema_peliosp  AAAGATTACAGACTTACCTACTATACCTGATTATGAGACCAAGAAACCGACATTTTAGCTGCATTTCCGCATGACTCCTCAGCTGGAGTTCACCTG
Conocephalum_92  AAAGATTATCGATTAACTTATTATACCTCCGATTATGAAACTAAAGATACGGATATTTTAGCTGCATTTAGATGACTCCTCAGCCTGGGTTACCAGCAG
Dumortiera_100   AAAGATTATCGATTAACTTATTACACTCCGATTATGATACCAAGGATACAGATATTTTGGCAGCCTTTAGATGACTCCTCAGCCTGGAGTACCAGCAG
Marchantia_5     AAAGATTATCGATTAACTTATTACACTCCGATTATGAGACCAAGGATACGGATATTTTAGCAGCATTTAGATGACTCCTCAGCCTGGAGTTCACCTG
Bazzania_jm      AAAGATTATAGATTAACTTATTACGCTGAATATGAGACCAAGAGACAGATATTTTGGCAGCATTTTCGTATGACTCCCCAACCGGGAGTACCACCTG
Metzgeria_3      AAAGATTACAGATTAAATTATTACACTCCAGATTATGAAACTAAAGATACAGATATTTTAGCAGCATTTTCGTATGACCCCTCAGCCTGGAGTACCAGAAG
Porella_4        AAAGATTATAGATCAACTTATTATACCTCCGACTATGAAACAAAGGAGACAGATATTTTAGCAGCATTTTCGTATGACTCCTCAACCTGGAGTACCAGAAG
Anthoceros_6     AAAGATTATAGATTAAACCATTTATACCTCGATTACGAGACCAAGGATACGTATATTTTGGCAGCGTCTTGAATGACTCCTTAACCAAGGGTGCCACCTG
Tetraphis_9      ?????????AGATTAACTTATTACACTCCAGATTATGAGACCAAGAGACCGATATTTTAGCAGCATTTTCGAATGACTCCTCAACCCGGAGTACCACCTG
Sphagnum_jm      AAAGATTACAGGTTGACTTATTACACCCGGAGTATGCTGTCAAAGATACCGACATTTTGGCAGCATTTTCGAATGACTCCTCAACCTGGAGTACCACCG
;
END;

```



## 5.9. The ‘Tools’ Menu

In addition to functionalities discussed above (the ‘*TreeViewer*’, section 5.6.1; the ‘*Ancestral states reconstruction*’ panel, section 5.6.2; and the ‘*Memory settings*’ window, Fig. 1a), the ‘*Tools*’ menu (Fig. 27a) also gives access to a ‘*Tree Generator*’, a ‘*Consensus Tree*’ builder, and a tool for computing pairwise distances. The ‘*Tree Generator*’ (Fig. 27b) allows for the generation of the NJ tree or any number of Loose Neighbor Joining (LNJ; section 5.4.3) or random trees. The trees generated are automatically transferred to the ‘*TreeViewer*’ under appropriate names (e.g., NJ, LNJ\_1, LNJ\_2, RANDOM\_1, RANDOM\_2).

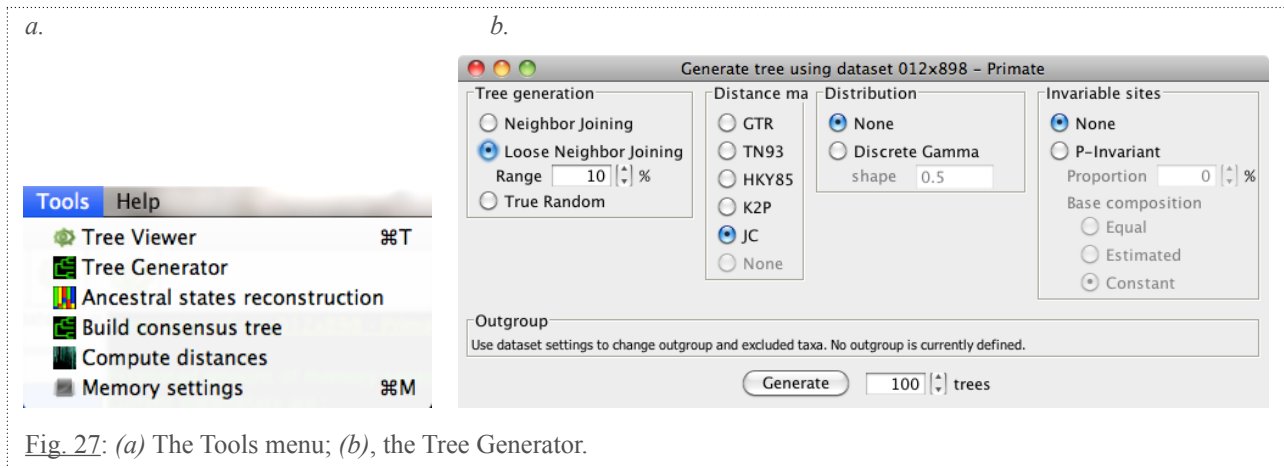


Fig. 27: (a) The Tools menu; (b), the Tree Generator.

In the ‘*Consensus tree builder*’ (Fig. 28a), trees in the left panel (corresponding to all trees from the ‘*TreeViewer*’) can be moved to the right panel for building a majority-rule consensus tree (with frequencies of clades) which is then automatically added to the *TreeViewer* under a chosen name (“my\_consensus\_tree” in Fig. 28a). The pairwise distances tool (Fig. 28b) allows for computing pairwise distances (among sequences of the active dataset) in the form of absolute numbers of differences or various distances: uncorrected (*none*) or corrected following a nucleotide or amino-acid or Codon substitution model with or without rate heterogeneity. Distances can be exported to a text file for spreadsheet applications such as Excel.

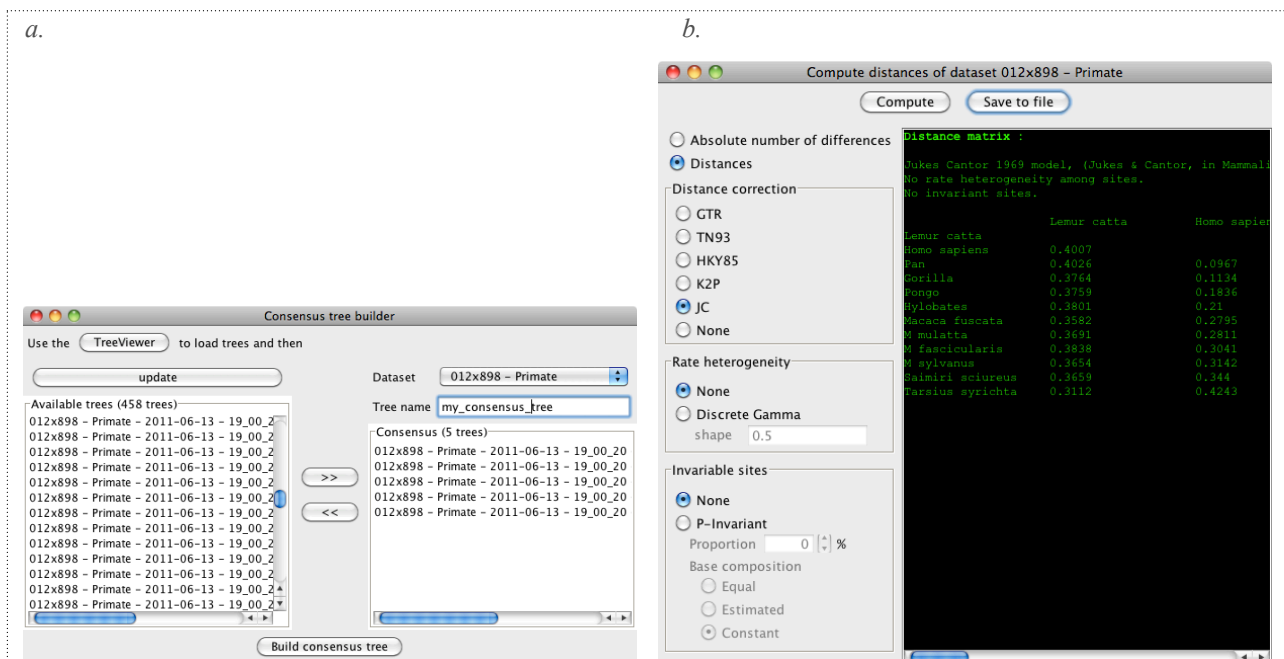


Fig. 28: (a) the Consensus tree builder; (b) the tool for computing pairwise distances.

## 5.10. Troubleshooting

Please, don't hesitate to contact us ([Dorde.Grbic@unige.ch](mailto:Dorde.Grbic@unige.ch) or [michel.milinkovitch@unige.ch](mailto:michel.milinkovitch@unige.ch)) if you encounter problems or bugs. We are also open to suggestions for improving the software.

A few problems that can arise when using MetaPIGA are listed below.

### Launching

When launching, MetaPIGA checks for the availability of updates (unless you have used the argument [noupdate] in command line). If you are connected to the internet, and there is no update to download, MetaPIGA will simply proceed with launching. If there is an update available, MetaPIGA will request your authorization to perform that update. If you are not correctly connected to the internet when launching the software, MetaPIGA will simply proceed with launching.

### Java errors at launch

- ✓ The Java 1.6 Virtual Machine (VM) must be installed on your computer for running MetaPIGA. If you only have earlier Java version(s) installed, your computer will complain, *e.g.*, with an error like that shown in Fig. 29. The Java 1.6 VM can be installed for Windows and Linux at <http://java.com/en/download/manual.jsp>. For Mac OSX, simply run the 'Software Update' feature available on the 'Apple menu'. To check, on your Mac, if Java 6 is installed and active, simply launch the 'Terminal.app' available in the "Utilities" sub-folder of the "Applications" folder. Then check your Java version by typing 'java -version', and pressing ENTER. If you are using the Snow Leopard Mac OS (OS X 10.6), you can check the version(s) of Java installed on your machine by launching the 'Java Preferences.app' available in the "Utilities" sub-folder of the "Applications" folder. Make sure that Java 6 (or later) is in the list AND active (*i.e.*, marked as in Fig. 30). You DON'T need to remove earlier Java versions (that might be required for older softwares). Note that if your Mac OS is older than 10.5, it will not support Java 1.6 ... hence, you will not be able to run MetaPIGA.

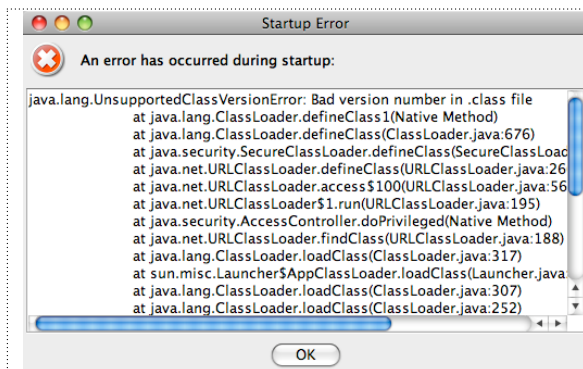


Fig. 29: Error message at launch due to the absence of a Java 1.6 (or later) VM.

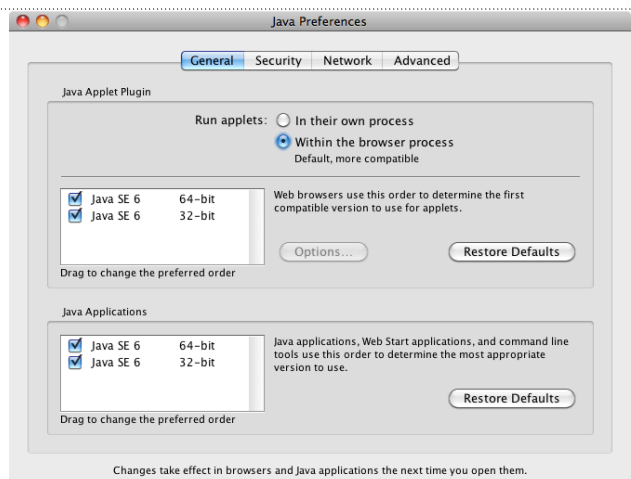


Fig. 30: The Java Preferences utility on Mac OS X.

- ✓ If MetaPIGA crashes at launch, it can also be due to a lack of memory. Try closing other applications, or change the maximum amount of memory allowed to MetaPIGA: in the file 'mp2\_console.vmoptions' (that you can find at the root of the MetaPIGA folder, *i.e.*, where the program is installed), set the Xmx value (and not the Xms value) to a lower value (expressed in megabytes; this value must be a multiple of 256). Note however that, to avoid problems, we made the installer allocate to MetaPIGA half of the memory available on your running machine. This should insure MetaPIGA to launch properly, even if other programs are running. Once



MetaPIGA has launched, the ‘*memory settings*’ (in the ‘*Tools*’ menu of MetaPIGA) allows changing the amount of memory allocated to MetaPIGA. The maximum value available in ‘*memory settings*’ is 1536 Mb on a 32-bit system (*i.e.*, the maximum allowed by java on such a system) ... even if the computer is equipped with more than 2Go of RAM. On the other hand, the maximum value available on a 64-bit system (*i.e.*, most of modern machines) can be much higher than 1536 Mb but is constrained to the amount of memory available on that machine *minus* 512Mb.

## Recovering results if a crash occurs

A ‘*Results.nex*’ file is written to the Results directory (see end of *section 5.4.5*) when the search is completed. On the other hand, the ‘*ConsensusTree.tre*’ file is automatically updated in the run directory during the search. Hence, if a crash occurs, for example after a significant running time involving a number of replicates, the ‘*ConsensusTree.tre*’ file (summarizing all replicates that accumulated before the crash) can be loaded and visualized in MetaPIGA after restarting. As the name of the tree includes the number of replicates, you will know when the crash occurred. As the consensus tree file is in Newick format, it can also be loaded in tree viewing softwares such as:

-FigTree (<http://tree.bio.ed.ac.uk/software/figtree/>)

-TreeView (<http://taxonomy.zoology.gla.ac.uk/rod/treeview.html>).

## Others

- ✓ When negative eigenvalues are encountered under GTR, an error message is generated and the search crashes.
- ✓ Sequences too dissimilar (>0.75 for DNA sequences, >0.95 for Protein sequences, and > 0.5 for standard binary data) can cause an error when computing distance matrices. The data quality control button (*i.e.*, ‘scissor’ button, *section 5.2.2*) and the ‘*check for saturation*’ function in the ‘*Dataset*’ menu allow avoiding that problem.

## 6. Acknowledgements

We are grateful to Alan Lemmon and Raphaël Helaers for continuing discussions on the metaGA. Funds were provided by the University of Geneva (Switzerland), the Swiss National Science Foundation, the Société Académique de Genève, the Georges and Antoine Claraz Foundation, the Ernst & Lucie Schmidheiny Foundation, and the ‘AAA/SWITCH – e-infrastructure for e-science program’. We thank Nabil Abdennadher and Mohamed Ben Belgacem (HES-Geneva) for assistance in the deployment of the Grid version of MetaPIGA. We thank Kim Roelants for designing the MetaPIGA logo (the flying pig .... because ... “*Pigs don’t fly, but MetaPIGA does*” ;-)

**Third party libraries:** MetaPIGA makes use of the following third party libraries (source code available through the corresponding links):

- The CERN Colt Scientific library 1.2.0 for pseudorandom number generation and statistics: <http://acs.lbl.gov/software/colt/>
- JAMA : A Java Matrix Package for matrix manipulations and eigen values decomposition: <http://math.nist.gov/javanumerics/jama/>
- The BioJava library to parse NEXUS files: <http://www.biojava.org/>  
*BioJava: an Open-Source Framework for Bioinformatics*  
*R.C.G. Holland; T. Down; M. Pocock; A. Prlić; D. Huen; K. James; S. Foisy; A. Dräger; A. Yates; M. Heuer; M.J. Schreiber*  
*Bioinformatics (2008) 24 (18): 2096-2097; doi: 10.1093/bioinformatics/btn397*
- The Google Collection classes library for its BiMap: <http://code.google.com/p/google-collections/>



## 7. Appendix 1: The MetaPIGA commands

MetaPIGA 2 requires only one thing to run: a nexus input file. This file must contain your sequence data following the standard Nexus data structure, i.e., using data blocks (or taxa + characters blocks). This file can be loaded and run either using the menu-driven interface (GUI) of MetaPIGA 2 or in command line.

All menus are described in detail above. However, the user can also choose to include all customized settings of MetaPIGA-2 in the Nexus input file and send it to the program for running without the use of the interface. This is particularly useful for performing unsupervised successive multiple long runs (batch files). In that case, the customized settings of MetaPIGA must be included in the Nexus input file in the form of a 'metapiga block'. The structure of this block is described hereafter. Note that, if you don't like typing the parameter settings yourself, you can use the MetaPIGA-2 user interface to generate, save, and run batch files.

### Conventions:

When a parameter is associated with information between round parentheses ( ), there must be no blank before or within the parentheses. For example:

```
DISTRIBUTION = GAMMA ( 4 )
```

cannot be written

```
DISTRIBUTION = GAMMA ( 4 )
```

nor

```
DISTRIBUTION = GAMMA ( 4 )
```

nor

```
DISTRIBUTION = GAMMA ( 4 )
```

All parameters between squared brackets [ ] are optional and can be omitted, and MetaPIGA will then use default values (underlined in the block description). Description of the commands is given in the above user manual (Section 5.3: 'Dataset settings').

## Batch files

You can easily create batch files, to run multiple analyses automatically. Batch files are nexus files in which you can add as many data block, metapiga blocks, and tree blocks as you wish. You must add a comment in the first line of each metapiga block in the form of a label using [BATCHLABEL = *label*]. Then, create a batch block that associates each run to (i) a data block and (ii) a metapiga block using those labels. The use of a tree block is optional.

For example, to run a given dataset with 2 different sets of parameters (the second requiring user-defined starting trees), the batch file will look like this:

```
BEGIN BATCH;
  RUN LABEL='run1' DATA=label1 PARAM=label1;
  RUN LABEL='run2' DATA=label1 PARAM=label2 TREE=label2;
END;
BEGIN METAPIGA; [BATCHLABEL = label1]
...
END;
BEGIN METAPIGA; [BATCHLABEL = label2]
...
END;
BEGIN DATA; [BATCHLABEL = label1]
...
END;
BEGIN TREE; [BATCHLABEL = label2]
...
END;
```



## BATCH Block

```
BEGIN BATCH;
  RUN LABEL='run_label' DATA=data_block_label
  PARAM=metapiга_block_label [TREES=tree_block_label];
  RUN ...
END;
```

## METAPIGA Block

```
BEGIN METAPIGA;
  [HEURISTIC
    'HC [RESTART = nbr_of_restart]
    | SA [COOLINGSCHEDULE = 'LUNDY | RP(delta) | CAUCHY | BOLTZMANN | GEOM(alpha)
    | LIN | TRI | POLY | EXP | LOG | PER | SPER | TANH | COSH'] [LunC = lundy_c]
    [LunALPHA = lundy_a] [INITACCEPT = value] [FINALACCEPT = value] [DELTA =
    'PERCENT[(p)] | BURNIN'] [REHEATING = 'DECREMENTS(d) | THRESHOLD(p) | NEVER']
    [COOLING = 'STEPS(steps) | SF(s,f)'] [DYNCS]
    | GA [NIND = individuals] [SELECTION = 'RANK | TOURNAMENT | REPLACEMENT[(s)]
    | IMPROVE | KEEPBEST'] [RECOMBINATION = rate] [OPERATORAPPLIEDTO = 'STEP |
    IND']
    | CP [CONSENSUS = 'STRICT | STOCHASTIC'] [OPERATOR = BLIND | SUPERVISED] [NPOP
    = populations] [NIND = individuals] [TOLERANCE = tolerance] [HYBRIDIZATION
    = rate] [SELECTION = 'RANK | TOURNAMENT | REPLACEMENT[(s)] | IMPROVE | KEEP-
    BEST'] [RECOMBINATION = rate] [OPERATORAPPLIEDTO = 'STEP | POP | IND'] [NCORE
    = cores]' ;]
  [EVALUATION [RATE = 'BRANCH | TREE'] [DATATYPE=CODON CODONRANGE{start_position-
    end_position}] [MODEL = 'GTR | TN93 | HKY85 | K2P | JC | GTR20 | WAG |
    JTT | DAYHOFF | VT | BLOSUM62 | CPREV | MTREV | RTREV | MTMAM | POISSON
    | GTR2 | ECM | GY'] [] [RATEPARAM {param(value) ...}] [AAFREQ = 'EMPIRICAL |
    ESTIMATED'] [DISTRIBUTION = 'NONE | GAMMA(subsets) | VDP(subsets)'] [DIST-
    SHAPE = shape] [PINV = proportion_of_invariant];]
  [SPECIFICPARTPARAM PARTNAME = charset-name [RATEPARAM {param(value) ...}] [DIST-
    SHAPE = shape] [PINV = proportion_of_invariant];]
  [OPTIMIZATION 'NEVER | CONSENSUSTREE | ENDSEARCH | DISC(s) | STOCH(p)' [ALGO =
    algorithm] [TARGET {param ...}] ;]
  [STARTINGTREE [GENERATION = 'NJ, LNJ(range), RANDOM, GIVEN'] [MODEL = 'GTR |
    TN93 | HKY85 | K2P | JC | GTR20 | POISSON | GTR2 | NONE'] [DISTRIBUTION =
    'NONE | GAMMA(shape) | VDP(subsets)'] [PINV = invariant] [PI = 'EQUAL |
    ESTIMATED | CONSTANT'];]
  [OPERATORS {operator[(parameter)] [operator[(parameter)] ...]} [SELECTION =
    'RANDOM | ORDERED | FREQLIST' ;]
  [FREQUENCIES {operator(frequency) ...} ;]
  [DYNAMICFREQ DYNOPERATORS {operator ...} [DINT = interval] [DMIN = minimum_fre-
    quency];]
  [SETTINGS [REMOVECOL = 'NONE | GAP | NGAP'] [DIR = 'output_directory'] [LABEL
    = 'run_label'] [GRID [SERVER = address] [CLIENT = id] [MODULE = id]];]
  [STOPAFTER [STEPS = steps] [TIME = hours] [AUTO = steps [AUTOTHRESHOLD =
    value]] [CONSENSUS [MRE = error] [GENERATION = steps] [INTERVAL = steps]]
    [NECESSARY {stop_condition ...}];]
  [REPLICATES [AUTOSTOP = 'NONE | MRE[(error)]] [RNUM = nbr_rep] [RMIN =
    nbr_rep] [RMAX = nbr_rep] [INTERVAL = interval] [PARALLEL = cores];]
  [OUTGROUP {taxa ...} ;]
  [DELETE {taxa ...} ;]
  [CHARSET NAME = charset-name SET{character-set ...} ;] ...
  [EXCLUDE {charset ...} ;]
  [PARTITION {charset ...} ;]
  [LOG {logFile ...} ;]
END;
```



## Description of the parameters:

1. **HEURISTIC** – By default, Metapiga uses the *metaGA* heuristic (*i.e.*, a genetic algorithm with consensus pruning; see Lemmon & Milinkovitch 2002 for details).

- **HC – Hill Climbing.** Tree space is explored using local perturbations (of topology and/or branch lengths and/or model parameters). New trees with improved likelihood are always accepted whereas trees with worse score are always discarded. This is the ‘stochastic hill climbing’ heuristic. We also implement a meta-heuristic called ‘random-restart hill climbing’. When the **RESTART** parameter is set to a value greater than 0, **RESTART+1** hill climbings are iteratively performed, each time with a different initial tree. Among the **RESTART+1** solution trees, only the best is kept.

Note that, when choosing the ‘Neighbor Joining’ starting-tree option (see **STARTINGTREE** parameter), the NJ tree will only be used for the first hill climbing, and Loose NJ trees will be generated for all restarts. Likewise, when choosing ‘user trees’ but the number of provided starting trees is smaller than **RESTART+1**, LNJ random trees will be generated for the missing starting trees. Note also that the stop conditions (see **STOPAF-TER** parameter) are defined for one hill climbing. For example, when choosing 10 restarts and ‘2000 steps’ as the stop condition, 11 hill climbing of 2000 steps will be performed, but only the best scored tree, among the 11 results, will be kept.

- **SA – Simulated Annealing.** Starting from a single tree, tree space is explored using local perturbations (of topology and/or branch lengths and/or model parameters). New trees with improved likelihood are always accepted, whereas trees with worse score are accepted with a probability which is a function of both the proportionate decrease in score and a control parameter called “*temperature*”. Much additional information is available in Kirkpatrick *et al.*, Optimization by Simulated Annealing, *Science*, 220, 4598, 671-680 (1983).

- **SCHEDULE** – The “cooling schedule” describes how the “*temperature*” decreases during the run.  $T$  is the temperature after decrements and  $N$  is the maximum number of temperature decrements before resetting the temperature to the starting temperature (see **REHEATING** parameter below). Except for the **LUNDY** cooling schedule, (and when it applies) are computed as follows: where  $\Delta$  is an upper bound on the change in likelihood,  $T_0$  is the initial and  $T_f$  the final acceptance parameters (see below). Available schedules are :

- **LUNDY** – The cooling schedule described by Lundy (1985).

with

being the parameter that controls the rate of cooling (its value is  $< 1$ ) where  $n$  is the number of sequences,  $(\text{taxa})$  is the number of sites, and  $\alpha$  and  $\beta$  are set between 0 and 1 (see **C** and **ALPHA** parameters below) and  $L$  is the log likelihood of the neighbor joining tree. It’s the default cooling schedule.

- **RP(delta)** – A ratio-percent cooling schedule.
- **CAUCHY** – Fast Cauchy schedule.
- **BOLTZMANN** – Boltzmann schedule.
- **GEOM(alpha)** – Geometric schedule.
- **LIN** – Linear schedule.
- **TRI** – Triangular schedule.
- **POLY** – Polynomial schedule.
- **EXP** – Transcendental (exponential) schedule.
- **LOG** – Transcendental (logarithmic) schedule.
- **PER** – Transcendental (periodic) schedule.
- **SPER** – Transcendental (smoothed periodic) schedule.
- **TANH** – Hyperbolic (tanh) schedule.
- **COSH** – Hyperbolic (cosh) schedule.

- **LUNC** – The parameter used in the **LUNDY** cooling schedule. You can set its value between [0,1] and the default value is 0.5.
- **LUNALPHA** – The parameter used in the **LUNDY** cooling schedule. You can set its value between [0,1] and the default value is 0.5.



- **INITACCEPT** – It's the initial maximum probability () to accept a tree with a 'worse' likelihood. Hence, it will define the starting temperature used when simulated annealing starts or when the temperature is reset (see **REHEATING** below). It's a probability, chosen between [0, 1], is set to 0.7 by default. Used with all cooling schedules except **LUNDY**.
- **FINALACCEPT** – It's the final maximum probability to accept a tree with a 'worse' likelihood (), so it will define the ending temperature used when simulated annealing should end or before resetting the temperature (see **REHEATING** below). It's a probability, chosen between [0, 1], must be smaller than **INITACCEPT** and is set to 0.01 by default. Only used with **LIN**, **TRI**, **POLY**, **EXP**, **LOG**, **PER**, **SPER**, **HYPTANH** and **HYPCOSH** cooling schedules.
- **DELTAL** – Determines how **is** initialized. **is** used to compute the starting temperature, and is the maximum distance between a current solution and a worse solution that could be accepted with a probability of **.**
  - **PERCENT** (*p*) – **is** set to a percentage *p* of the Neighbor Joining Tree log likelihood. You can set the value of *p* between [0,1] and the default is 0.001 (0.1% of the NJT).
  - **BURNIN** – Selected operators are used on the starting tree for a burn-in period of 20 applications for each operator. The maximum change in log likelihood observed during this period is used as **.**
- **REHEATING** – Determines under which condition the temperature is reset to the initial starting temperature.
  - **NEVER** – Temperature is never reset, but this option can only be selected with **LUNDY**, **RP**, **CAUCHY**, **BOLTZMANN** and **GEOMETRIC** cooling schedules.
  - **DECREMENTS** (*d*) – Temperature is reset when it has decreased *d* times. It's the default **REHEATING** option, usable with all cooling schedules.
  - **THRESHOLD** (*p*) – Temperature is reset when it attains a threshold equal to **Note that** **must** be smaller than 1 and sufficiently small (0.001 is the default value). This **REHEATING** option can only be used with **LUNDY**, **RP**, **CAUCHY**, **BOLTZMANN** and **GEOMETRIC** cooling schedules.
- **COOLING** – Establishes the number of times a tree is modified before the temperature is decreased. You can choose between 2 cooling types:
  - **STEPS** (*steps*) – Stay at the same temperature for a given number of steps.
  - **SF** (*s*, *f*) – Lower the temperature after *s* successes or *f* failures, whichever comes first. Successes are tree modifications that improve the likelihood and failures are those that do not. This **COOLING** is used by default, with *s*=10 and *f*=100.
- **GA – Genetic Algorithm** . At each step (generation) of the heuristic, each individual of a population of trees is mutated using the selected operators. Death / survival of individuals is controlled using a selection scheme.
  - **NIND** – The number of individuals (trees) within the population (set). Set to 8 by default.
  - **SELECTION** – The method used to control death / survival of individuals :
    - **RANK** – We implement a rank selection similar to that described in (Lewis 1998, Mol. Biol. Evol. 15, 277-283). The individual having the highest lnL is automatically allowed to leave  $k=0.25*NIND$  offspring (i.e., copies of itself) in the next generation. Then, each individual is assigned a probability *p* of leaving an offspring as a function of its position in a list in which individuals are ranked by their score. The probability *p* for the *i*th individual of leaving an offspring to the next generation is equal to:
    - **TOURNAMENT** – Two individuals are drawn randomly from the population of *n* individuals, and one offspring is produced from the individual with higher score. Both trees are then placed back into the mating population, and the whole process is repeated until *n* offspring have been generated. This is the default selection method.
    - **REPLACEMENT** – Two individuals are drawn randomly from the population of *n* individuals and two copies of the better individual are returned to the mating pool (parents are discarded). The process is repeated *sn* times, where *s* is the strength of the selection (1.0 by default), then the offspring population is generated as an exact copy of the post-selection parent population.
    - **IMPROVE** – Only those individuals that have scores better than that of the best tree from the previous generation are kept. Each individual that fails this test is discarded and replaced by a copy of the current best individual.



- **KEEPBEST** – Only the best individual of each population is kept, others are replaced by a copy of it.
- **RECOMBINATION** – Each counter-selected sub-optimal individual has a probability  $p$  (between  $[0, 1]$  and set to 0.1 by default) to recombine with a better individual in the population. Recombination is performed by exchanging subtrees defined by one of the identical taxa partitions in the two parental trees (i.e., one internal branch that defines subtrees including the same taxa but with potentially different sub-topologies). If no common branch exists, the offspring is defined as a copy of the best individual. A recombination event can be viewed as a large number of simultaneous topological mutations. The exact procedure depends on the selection scheme:
  - **RANK** – Recombination is not available under that selection scheme.
  - **TOURNAMENT** – With a probability  $p$ , the offspring set after a tournament is not a copy the individual with higher score but a recombination between the two trees that have been initially drawn for tournament.
  - **REPLACEMENT** – With a probability  $p$ , only one (instead of two) copy of the better individual is returned to the mating pool. The second individual returned is a recombination between the two trees that have been initially drawn.
  - **IMPROVE** – Each individual that does not have a score better than that of the best tree from the previous generation has a probability  $p$  of leaving an offspring by recombining with the current best individual.
  - **KEEPBEST** – Each individual that does not have a score better than that of the best current individual has a probability  $p$  of leaving an offspring by recombining with the current best individual.
- **OPERATORAPPLIEDTO** – **IND** is the default
  - **STEP** – At each step of the heuristic, a single mutation operator is selected and applied to each tree of each population.
  - **IND** – At each step of the heuristic, each individual is separately assigned a mutation operator.
- **CP – Consensus pruning (MetaGA)**. This is the core of the "metaPopulation genetic Algorithm" (Lemmon & Milinkovitch, *PNAS* 99:10516-10521 (2002)):  $P$  sets (populations) containing each  $I$  trees (individuals) are forced to cooperate in the search for the optimal trees. At each step (generation) of the heuristic, individuals are mutated following inter-populations consensus rules. Death / survival of individuals is defined using a selection scheme.
  - **CONSENSUS** – **STOCHASTIC** is chosen by default
    - **STRICT** – Any branch shared by all trees across all populations (100% consensus) will not be mutated. Mutations on any other branch will be unconstrained.
    - **STOCHASTIC** – Each branch (partition) common to at least two trees will be assigned a consensus value. The probability of any mutation affecting that partition is  $1 - (\text{consensus value})$ . Example: if a given branch is shared by 12 among 16 trees (e.g., 4 populations of 4 individuals each), any mutation affecting that branch will be accepted with a probability of 0.25. A branch shared by all trees will never be mutated.
  - **OPERATOR** – If operator is set to **BLIND**, a mutation breaking a consensus won't be applied and the tree will remain unchanged until the next mutation (at generation  $i+1$ ). If operator is set to **SUPERVISED**, the operator will search for candidate mutations that don't break any consensus. If no such candidate exists, no mutation is performed and the tree will remain unchanged until the next generation.
  - **NPOP** – The number of populations (sets). Set to 4 by default.
  - **NIND** – The number of individuals (trees) within each population (set). Set to 4 by default.
  - **TOLERANCE** – The **CONSENSUS** command constrains how shared branches are modified. The **TOLERANCE** parameter avoids partitions to become "frozen", i.e., inaccessible to mutations. The **TOLERANCE** parameter helps avoiding to be trapped in a possible local optimum. Set to 0.5 by default. Example: With "strict consensus" and a tolerance of 0.1, any branch shared by all trees is anyway mutated with a probability of 0.1.
  - **HYBRIDIZATION** – At each generation, there is a probability (between  $[0, 1]$  and set to 0.1 by default) that all sub-optimal individuals from one random population are not mutated but, instead, are

recombined with one individual from another population; sub-optimal individuals from other populations experience the normal mutation procedure.

- **SELECTION** – The method used to control death / survival of individuals :
  - **RANK** – We implement a rank selection similar to that described in (Lewis 1998, Mol. Biol. Evol. 15, 277-283). The individual having the highest InL is automatically allowed to leave  $k=0.25*N_{IND}$  offspring (i.e., copies of itself) in the next generation. Then, each individual is assigned a probability  $p$  of leaving an offspring as a function of its position in a list in which individuals are ranked by their score. The probability  $p$  for the  $i$ th individual of leaving an offspring to the next generation is equal to:
  - **TOURNAMENT** – Two individuals are drawn randomly from the population of  $n$  individuals, and one offspring is produced from the individual with higher score. Both trees are then placed back into the mating population, and the whole process is repeated until  $n$  offspring have been generated.
  - **REPLACEMENT** – Two individuals are drawn randomly from the population of  $n$  individuals and two copies of the better individual are returned to the mating pool (parents are discarded). The process is repeated  $sn$  times, where  $s$  is the strength of the selection (1.0 by default), then the offspring population is generated as an exact copy of the post-selection parent population.
  - **IMPROVE** – Only those individuals that have scores better than that of the best tree from the previous generation are kept. Each individual that fails this test is discarded and replaced by a copy of the current best individual. This is the default selection method.
  - **KEEPBEST** – Only the best individual of each population is kept, others are replaced by a copy of it.
- **RECOMBINATION** – Each counter-selected sub-optimal individual has a probability  $p$  (between [0, 1] and set to 0.1 by default) to recombine with a better individual in the population. Recombination is performed by exchanging subtrees defined by one of the identical taxa partitions in the two parental trees (i.e., one internal branch that defines subtrees including the same taxa but with potentially different sub-topologies). If no common branch exists, the offspring is defined as a copy of the best individual. A recombination event can be viewed as a large number of simultaneous topological mutations. The exact procedure depends on the selection scheme:
  - **RANK** – Recombination is not available under that selection scheme.
  - **TOURNAMENT** – With a probability  $p$ , the offspring set after a tournament is not a copy the individual with higher score but a recombination between the two trees that have been initially drawn for tournament.
  - **REPLACEMENT** – With a probability  $p$ , only one (instead of two) copy of the better individual is returned to the mating pool. The second individual returned is a recombination between the two trees that have been initially drawn.
  - **IMPROVE** – Each individual that does not have a score better than that of the best tree from the previous generation has a probability  $p$  of leaving an offspring by recombining with the current best individual.
  - **KEEPBEST** – Each individual that does not have a score better than that of the best current individual has a probability  $p$  of leaving an offspring by recombining with the current best individual.
- **NCORE** – The number of cores/processors assigned for parallel processing. Different populations will be assigned to different cores. Set to 1 by default (no parallelization). **WARNING:** this parameter should be considered in combination with the `PARALLEL` parameter (in `REPLICATES`). It is advised to leave the `NCORE` parameter to 1 when you perform replicates with parallelization.
- **OPERATORAPPLIEDTO** – `IND` is the default
  - **STEP** – At each step of the heuristic, a single mutation operator is selected and applied to each tree of each population .
  - **POP** – At each step of the heuristic, each population is separately assigned a mutation operator (i.e., that operator is applied to all individuals within a population).
  - **IND** – At each step of the heuristic, each individual is separately assigned a mutation operator .

2. **EVALUATION** – By default, MetaPIGA evaluates trees with the maximum likelihood criterion using a single rate matrix *R* for the **TREE**, the **JC** model and no rate heterogeneity. Note that if the dataset is partitioned with charsets, some parameters (**RATEPARAM**, **DISTSHAPE**, **PINV**) can be overridden with the **SPECIFICPARTPARAM** command for each partition.
  - **RATE** – The rate matrix *R* (by default, MetaPIGA use one *R* for the **TREE**):
    - **BRANCH** – NOT AVAILABLE YET – A different rate matrix *R* is used for each branch.
    - **TREE** – A single rate matrix *R* is used across the whole tree.
  - **DATATYPE=CODON** – This token defines that nucleotides in this data set should be interpreted as codons. The token has to be followed by the **CODONRANGE** {*first\_position*–*last\_position*} token, where '*first\_position*' and '*last\_position*' define the range of nucleotide indexes that will be interpreted as codons.
  - **MODEL** – Depending on the datatype (DNA or PROTEIN or STANDARD), the default substitution model is **JC**, **POISSON**, or **GTR2**, respectively. You can set substitution models with:
    - **GTR** – General-Time-Reversible model for nucleotides.
    - **HKY85** – Hasegawa-Kishino-Yano 1985 model (nucleotides).
    - **TN93** – Tamura-Nei 1993 model (nucleotides).
    - **K2P** – Kimura's 2 Parameter model (nucleotides).
    - **JC** – Jukes Cantor 1969 model (nucleotides).
    - **GTR20** – General-Time-Reversible model for proteins.
    - **WAG** – Wheland and Goldman model (proteins).
    - **JTT** – Jones-Taylor-Thornton model (proteins).
    - **DAYHOFF** – Dayhoff model (proteins).
    - **VT** – Variable Time substitution matrix (proteins).
    - **BLOSUM62** – BLOSUM62 (BLOCKs of amino acid SUBstitution Matrix) substitution matrix (proteins).
    - **CPREV** – Chloroplast reversible substitution model (proteins).
    - **MTREV** – Reversible mitochondrial substitution model (proteins).
    - **RTREV** – RtREV substitution matrix (proteins).
    - **MTMAM** – Mtmam model (for mitochondrial data) (proteins).
    - **POISSON** – Poisson model (proteins).
    - **GTR2** – General-Time-Reversible model for standard binary data.
    - **ECM** – Empirical codon model for codon data.
    - **GY** – Goldman-Yang model for codon data.
  - **RATEPARAM** – Set the values of each parameter of the rate matrix *R*.
    - **A | B | C | D | E** – The five parameters that can be set with GTR. Set to 0.5 by default.
    - **K** – The kappa parameter of K2P and HKY85. Set to 0.5 by default.
    - **K1 | K2** – The 2 parameters of TN93 (respectively **K1** are transitions between purines, and **K2** transitions between pyrimidines). Set to 0.5 by default.
    - **AR | AN | AD | ... | WY | WV | YV** – The 189 parameters that can be set for GTR20. They correspond to the upper right triangle of the GTR substitution matrix, with the 20 amino acids ordered by alphabetical order of their 3-letter names (A R N D C Q E G H I L K M F P S T W Y V). For example, A<->R rate is set using AR parameter (RA will not be recognized). Set to 0.5 by default.
  - **AAFREQ** – Used for empirical protein models with unequal equilibrium state frequencies (**EMPIRICAL** by default).
    - **EMPIRICAL** – Equilibrium amino-acid frequencies are fixed to the empirical values reflecting estimates of the corresponding model.
    - **ESTIMATED** – Equilibrium amino-acid frequencies are fixed to those observed in the dataset.
  - **DISTRIBUTION** – The rate heterogeneity (none by default).
    - **NONE** – No rate heterogeneity
    - **GAMMA** – Rate heterogeneity following a Gamma distribution. The number of rate categories (4 by default) and shape parameter alpha (default=1) can be defined.
  - **DISTSHAPE** – Shape parameter (alpha) of the gamma distribution. Set to 1.0 by default.
  - **PINV** – Proportion of invariable sites (between 0 and 1). Set to 0 (no invariant) by default.
3. **SPECIFICPARTPARAM** – Specific evaluation parameters can be set for each charset separately if the dataset is partitioned. If no **SPECIFICPARTPARAM** is defined for a given partition, parameters defined with the **EVALUATION** command will be used.



- **PARTNAME** – The name of the partition to which the parameters apply. ATTENTION: the partition name must always be defined before **RATEPARAM**, **DISTSHAPE** and **PINV**.
  - **RATEPARAM** – Set the value of each parameter of the rate matrix R.
    - **A | B | C | D | E** – The five parameters that can be set with GTR. Set to 0.5 by default.
    - **K** – The kappa parameter of K2P and HKY85. Set to 0.5 by default.
    - **K1 | K2** – The 2 parameters of TN93 (respectively **K1** are transitions between purines, and **K2** transitions between pyrimidines). Set to 0.5 by default.
    - **AR | AN | AD | ... | WY | WV | YV** – The 189 parameters that can be set for GTR20. They correspond to the upper right triangle of the GTR substitution matrix, with the 20 amino acids ordered by alphabetical order of their 3-letter names (A R N D C Q E G H I L K M F P S T W Y V). For example, A<->R rate is set using AR parameter (RA will not be recognized). Set to 0.5 by default.
  - **DISTSHAPE** – Shape parameter (alpha) of the gamma distribution. Set to 1.0 by default.
  - **PINV** – Proportion of invariable sites (between 0 and 1). Set to 0 (no invariant) by default.
4. **OPTIMIZATION** – For configuring intra-step optimization frequencies, algorithm and targets. There are 5 ways of choosing when MetaPIGA optimizes the tree during a heuristic. With **NEVER**, no optimization algorithm is applied. With **ENDSEARCH**, final trees are optimized at the end of the heuristic. With **CONSENSUSTREE**, only the final consensus tree built using all replicates is optimized (when performing single searches, *i.e.* one single replicate, no consensus is built and no intra-step optimization of target parameters is performed). With **STOCH** (*p*), with *p* between [0.01, 1], there is a probability *p* at each step to optimize the tree. With **DISC** (*s*), trees will be optimized every *s* steps. Note that (1) with **STOCH** and **DISC**, optimization of the final trees is also performed at the end of the heuristic (hence, at the end of each replicate if multiple replicates are performed) and (2) with **END-SEARCH**, **STOCH** and **DISC**, the final consensus tree is also optimized when multiple replicates are performed. You can also set :
- **ALGO** – Set the algorithm used for intra-step optimization.
    - **GA** – Genetic algorithm. Simple GA without recombination: each tree to be optimized is copied 7 times and that population of 8 individuals is experiencing mutations (of targets, see below). Selection is performed with **IMPROVE** (see above). The GA is stopped when the likelihood remains unchanged for 200 steps (generations).
    - **POWELL** – NOT AVAILABLE YET – Direction set (Powell's) method in multidimensions, using golden section search to bracket a minimum of the likelihood function, and Brent's method to isolate the minimum.
    - **DFO** – NOT AVAILABLE YET – Derivative-Free Optimization. The method used is a trust-region algorithm that employs interpolation models of degree at most 2 to build a model of the objective function. The models are constructed using Newton fundamental polynomials.
  - **TARGET** – Set the targets of the optimization procedure.
    - **BL** – Branch lengths.
    - **R** – Parameter(s) of the rate matrix R (not relevant with Jukes Cantor model).
    - **GAMMA** – Shape parameter *alpha* of the gamma distribution (only relevant when rate heterogeneity is used).
    - **PINV** – Proportion of invariable sites (only relevant when invariant sites are used).
    - **APRATE** – Among-Partition rate variation (relative branch lengths are only relevant when the data-set is partitioned into charsets).
5. **STARTINGTREE** – Method used to generate the starting tree(s) for the heuristic. When using starting trees generated by NK or LNJ (see below), a model (and potentially rate heterogeneity distribution and proportion of invariable sites) must also be set for computing the distance matrix.
- **GENERATION** – By default, MetaPIGA uses Loose Neighbor Joining Trees (LNJ) as starting trees.
    - **NJ** – Starting trees are built using the Neighbor Joining method (Saitou & Nei 1987).
    - **LNJ** (*range*) – Loose Neighbor Joining. Range is a percentage value, that must be greater than 0 and smaller than 1. Starting trees have pseudo-random topologies based on the Neighbor Joining algorithm. The classical NJ method joins 2 nodes having minimal rate-corrected distance. Here, under LNJ, a list containing the  $(range \times (NTax \times NTax - 1) / 2)$  smaller distances will be built and two nodes will be randomly selected from it. Branch lengths are computed normally using the Neighbor Joining method (Saitou & Nei 1987). If the *range* parameter is close to 0, the LNJ tree will be similar to the neighbor joining tree; if it's close to 1, the tree will exhibit essentially a random topology.



- **RANDOM** – Starting trees have random topologies and random branch lengths. No distance matrix is used, so you can't choose a substitution model or rate distribution or proportion of invariable sites. The random topology is generated by starting with a "root" node with three branches ending each with an 'open slot'. We know the list of available taxa and we know that the number of internal nodes in the final tree will be (T-2-root). The tree generator cycles through the list of open slots. Each time an open slot is visited, there is a probability  $p=0.5$  to fill the slot either with one of the available taxa or with one of the available internal nodes (connected to two new branches, each ending with an open slot). An internal node is always added if only one open slot remains. The algorithm stops when all internal nodes and taxa have been incorporated. Branch lengths are drawn from an exponential distribution (with  $\lambda=1$ ), and shifted by 0.001 (such that the minimum value is 0.001 and the mean is 1.001).
  - **GIVEN** – User tree(s). If your NEXUS file contains a TREE block (and the command GIVEN is used), and if you selected SA or HC as the heuristic option, the first tree in the tree block will be loaded and used as starting tree. If you selected CP as the heuristic option with NPOP populations, the NPOP first trees in the TREE block will be loaded (one tree per population). If you selected GA as the heuristic option, the NIND first trees in the TREE block will be loaded (one tree per individual). More options for importing user starting trees are available in the GUI (see point 5.3.4. in the manual above).
  - **MODEL** – Depending on the datatype (DNA or PROTEIN or STANDARD), the default substitution model to generate distance matrices is JC, Poisson, or GTR2, respectively. You can set substitution models with :
    - **GTR** – General-Time-Reversible model for nucleotides.
    - **HKY85** – Hasegawa-Kishino-Yano 1985 model (nucleotides).
    - **TN93** – Tamura-Nei 1993 model (nucleotides).
    - **K2P** – Kimura's 2 Parameter model (nucleotides).
    - **JC** – Jukes Cantor 1969 model (nucleotides).
    - **GTR20** – General-Time-Reversible model for proteins.
    - **Poisson** – Poisson model (proteins).
    - **GTR2** – General-Time-Reversible model for standard binary data.
    - **NONE** – No distance matrix.
  - **DISTRIBUTION** – The rate heterogeneity (none by default)
    - **NONE** – No rate heterogeneity.
    - **GAMMA** – Rate heterogeneity following a Gamma distribution. The number of rate categories is fixed to 4 but the shape parameter alpha (default=0.5) can be defined.
  - **PINV** – Proportion of invariable sites (between 0 and 1). Set to 0 (no invariant) by default. If  $P_{INV} > 0$ , the total number of sites is adjusted to have distances equal to the mean number of substitutions over variable sites only.
  - **PI** – Base composition of invariant sites (used only if  $P_{INV} > 0$ ).
    - **EQUAL** – The invariant sites will have base composition equal to 0.25.
    - **ESTIMATED** – The invariant sites base composition is set to the average base composition across all sequences.
    - **CONSTANT** – (Default) The invariant sites base composition is set to the average base composition of the site which are constant.
6. **OPERATORS** – Sets the operators used to generate new solution trees. You can list more than one operator, and some can have specific parameters.
- **SELECTION** This keyword can be set to:
    - **ORDERED** – Selected operators are chosen one after another.
    - **RANDOM** – (Default) Selected operators are randomly drawn.
    - **FREQLIST** – Selected operators are drawn following probabilities defined in FREQUENCIES.
  - If OPERATORS parameter is not set, MetaPIGA uses the following operators by default: NNI, BLMINT, TXS(2), STS(2). Available operators are:
    - **NNI** (NEAREST-NEIGHBOR INTERCHANGE) – Two grand-children branches of a random internal node are swapped.
    - **SPR** (SUBTREE PRUNING AND REGRAFTING) – Removes a branch from the tree with a subtree attached to it and re-grafts the subtree elsewhere.
    - **TBR** (TREE-BISECTION-RECONNECTION) – Breaks a branch and reconnects each of the two subtrees on a random branch.

- **TXS** (**TAXA SWAP**) – Swaps a given number of randomly-chosen leaves (defined between parentheses). The value for this operator is a number between 2 and the number of leaves. You can also set the parameter to **ALL** (swap all leaves) or **RANDOM** (swap a random number of leaves). If you set a number smaller than 2, 2 leaves will be permuted. If you set a number greater than the number of leaves, **ALL** leaves will be permuted. Default parameter is 2.
- **STS** (**SUBTREE SWAP**) – By default, swaps two randomly-chosen internal nodes (*i.e.*, subtrees that contain more than one leaf). If the parameter is set to **RANDOM** instead of 2, the whole tree will be divided into a random number of subtrees, and all of them will be permuted.
- **BLM** (**BRANCH LENGTH MUTATION**) – Randomly changes the length of a randomly-chosen branch by multiplying the parameter's value of the previous generation by a random number drawn from an exponential distribution (with  $\lambda=2$ ), and shifted by 0.5 (such that the minimum value is 0.5 and the mean is 1).
- **BLMINT** (**BRANCH LENGTH MUTATION ONLY ON INTERNAL BRANCHES**) – Randomly changes the length of a randomly-chosen internal branch by multiplying the parameter's value of the previous generation by a random number drawn from an exponential distribution (with  $\lambda=2$ ), and shifted by 0.5 (such that the minimum value is 0.5 and the mean is 1).
- **RPM** (**RATE PARAMETERS MUTATION**) – Randomly changes the R matrix values by multiplying the value of the previous generation by a random number drawn from an exponential distribution (with  $\lambda=2$ ), and shifted by 0.5 (such that the minimum value is 0.5 and the mean is 1). Parameter for this operator is the number of R elements to change (1 or **ALL**).
- **GDM** (**GAMMA DISTRIBUTION MUTATION**) – Randomly changes the alpha parameter of the Gamma distribution by multiplying the parameter's value of the previous generation by a random number drawn from an exponential distribution (with  $\lambda=2$ ), and shifted by 0.5 (such that the minimum value is 0.5 and the mean is 1). Only available when gamma-distribution rate heterogeneity has been selected.
- **PIM** (**PROPORTION OF INVARIANT MUTATION**) – Randomly changes the proportion of invariable sites by multiplying the parameter's value of the previous generation by a random number drawn from a normal distribution (with mean=1 and SD= 0.5). The resulting multiplier is rejected if  $\leq 0.4$ . Only available when proportion of invariable sites has been selected.
- **APRM** (**AMONG-PARTITION RATE MUTATION**) – Randomly changes the among-partition rates for relative branch lengths by multiplying the parameter's value of the previous generation by a random number drawn from a normal distribution (with mean=1 and SD= 0.5). The resulting multiplier is rejected if  $\leq 0.4$ . Only available when the dataset is partitioned with "charsets".

7. **FREQUENCIES** – Used to set the frequencies of operators, using *operator (frequency)* .

8. **DYNAMICFREQ** – Operators set to *dynamic* have their probabilities of use automatically adjusted at every 'interval' to reflect their relative contributions to score improvements (the probability of using a specific operator is increased or decreased, if its contribution to the score improvement is increased or decreased, respectively). You can set some parameters for dynamic frequencies :

- **DYNOPERATORS** – The list of operators is set to dynamic.
- **DINT** – Interval (in number of steps) used to recompute the frequencies. Set to 100 by default.
- **DMIN** – Frequencies can't be decreased under the lower bond . Set to 0.04 by default.

9. **SETTINGS** – Some miscellaneous MetaPIGA settings

- **REMOVECOL** – Set to **NONE** by default, treating gaps ('-') as **N** (A or C or T or G) in nucleotide datasets, or as **X** (any amino acid) in protein datasets, or as **?** (0 or 1) in standard datasets. Can be set either to **GAP**, for removing every column containing a gap ('-'), or to **NGAP** for removing every column containing a gap or a N/X/? in nucleotide/protein/standard datasets.
- **DIR** – Defines the whole path where the Results folder will be placed. By default, results folders are put in a 'MetaPIGA results' folder in your home directory (*e.g.* 'My documents' in Windows). If you use the **DIR** command in a Nexus file, you **MUST** put the folder name between quotes.
- **LABEL** – Defines the name of the Results folder for output files. Changing the label changes the Results folder name but not the nexus file name. The Results folder will be placed into the directory defined with the **DIR** command. The Result folder is named with its label followed by the date (year-month-day) and followed by the time (hour\_min\_sec) at the which the search was started. This allows for easy differentiation of results performed at different times on the same dataset. If you use the **LABEL** command in a Nexus file, you **MUST** put the label name between quotes.



- **GRID** – NOT AVAILABLE YET – MetaPIGA will run through a GRID using the XtremWeb-CH middleware (see <http://www.xtremwebch.net/>). You must specify the server address (e.g. **SERVER=HTTP://ADDRESS:8080**), your identifier on the GRID (**CLIENT** command) and the identifier of the MetaPIGA module on the GRID (**MODULE** command). Note that when MetaPIGA runs on a GRID, it does not generate any log file (**LOG** command is ignored). GRID running is disabled by default.
- 10. **OUTGROUP** – Sets any number of taxa that will form the outgroup (all other taxa are in the ingroup). Operators will never mix up taxa between the outgroup and the ingroup. The tree is rooted between outgroup and ingroup.
- 11. **DELETE** – Sets any number of taxa that will be removed from the analysis.
- 12. **CHARSET** – Defines a charset ; you must use a different **CHARSET** command for each charset to be defined. For each one, you must give its **NAME** and a list of character positions with **SET**. For defining a range of character positions, you can use 2 positions separated by ‘-’ (like 60–125), and potentially add ‘/’ and the interval size. For example 60–125/3 will take positions 60, 63, 66, 69, 72, ..., 120, 123. Charsets can be defined as the combination of other charsets (defined higher in the **METAPIGA** block) or by the combination of charset(s) and character list.
- 13. **EXCLUDE** – Sets any number of charsets that will be excluded from the analysis. A charset is defined by 2 character positions (like 60–125), or can be defined with the **CHARSET** command.
- 14. **PARTITION** – Divides the data matrix in charsets and compute likelihood separately for each charset. A charset is defined by 2 character positions (like 60–125), or can be defined with the **CHARSET** command.
- 15. **STOPAFTER** – Sets the stop criterion of the heuristic. Any number of conditions can be set and each one can be necessary or sufficient. The heuristic stops when any of the sufficient conditions is met **or** when **all** necessary conditions are met. Conditions are sufficient by default and can be switch to necessary using the **NECESSARY** command. If **STOP AFTER** is not set, the heuristic will not start but starting tree(s) will be generated.
  - **STEPS** – Defines a maximum number of generations.
  - **TIME** – Allows to stop the heuristic after a given amount of time (in hours).
  - **AUTO** – **AUTO** will stop the heuristic if the best solution evaluation doesn’t improve more than a given percentage (**AUTO\_THRESHOLD** parameter, set to 0.0001 by default, *i.e.* 0.01%) at any step during the defined number of steps.
  - **CONSENSUS** – **CONSENSUS** can only be used with Consensus Pruning (metaGA heuristic), and will stop the heuristic when the mean relative error among consensus trees (**INTERVAL** parameter, set to 10 by default) remains below a given value (set with **MRE** parameter, 0.03 by default). Each consensus tree is built using all trees from all populations in a generation. As consensus trees tend not to vary much between 2 consecutive generations, the user is advised to allow several generations between sampling (with **GENERATION** parameter, set to 5 by default).
  - **NECESSARY** – The following conditions can be switched to necessary : **STEPS**, **TIME**, **AUTO**, **CONSENSUS**.
- 16. **REPLICATES** – The number of times the metaheuristic will be repeated with the same dataset. At the end, a majority-rule consensus tree is produced. By default, only one tree is produced.
  - **AUTOSTOP** – Adds a stop condition to replicates’ generation.
    - **NONE** – By default, there is no stop condition, so a given number of replicates is produced. You can set the number of replicates produced with **RNUM** parameter.
    - **MRE (error)** – This option allows MetaPIGA to stop producing replicates when the Mean Relative Error among consecutive consensus trees remains below a given value. *Error* is a value between [0,1] set to 0.05 by default.
      - **RMIN** – The minimum number of replicates to produce. Default value is 100.
      - **RMAX** – The maximum number of replicates to produce. Default value is 10 000.
      - **INTERVAL** – The number of consecutive consensus trees (set to 10 by default) that must have a MRE below a given value before stopping the production of replicates .
  - **PARALLEL** – The number of replicates to be run in parallel (*i.e.*, simultaneously). By default, this parameter is set to 1 (no parallel processing). **WARNING**: It is strongly advised not to use a value greater than the number of processors/cores available on the running computer. **WARNING2**: this parameter must be considered in combination with the parameter **NCORE** (*i.e.*, the number of cores/processors assigned for parallel processing **WITHIN** a replicate). For example, if you use a computer with 4 cores, set the **NCORE** parameter to 1 and the **PARALLEL** parameter to 4, such that each replicate will use a single core (*i.e.*, 4 replicates will be run simultaneously). If you use a computer with 8 cores, you can set the **NCORE** parameter to 2 and

the `PARALLEL` parameter to 4, such that each replicate will use two cores AND 4 replicates will be run simultaneously.

17. **Log** – Set the log files you want as output. They can give you valuable information on what happens during the execution of MetaPIGA. Be aware that selecting the log files indicated with asterisks can (i) significantly slow down the search and (ii) fill up large amount of disk space (with the magnitude of slow-down / fill-up approximately indicated by the number of asterisks). All log files are written in the results folder.
- **DATA** – Working matrix log file - Prints the compressed dataset to '*Dataset.log*'. The last row contains the weight of each column, i.e., the number of times this data pattern is found in the data matrix. .
  - **DIST** – Distance matrix log file - Prints the distance matrix to '*Distances.log*'.
  - **TREESTART** – Starting Trees log file - Prints the starting tree(s) to '*StartingTrees.tre*'.
  - **HEUR (\*)** – Heuristic search log file - The '*Heuristic.log*' file records details about each step of the heuristic used. It requires disk space between 500 bytes and 1 Kb per iteration of the heuristic.
  - **TREEHEUR (\*\*)** – Heuristic search tree file - The '*Heuristic.tre*' file records each tree found at each step of the heuristic. It requires disk space of +/- 130 bytes per taxa per tree recorded. For example, recording trees for a dataset of 200 taxa, using the metaGA heuristic with 4 populations of 4 individuals each, for a fixed amount of 5000 generations will generate a file of about 1.5Gb for each replicate produced.
  - **CONSENSUS (\*\*)** – Consensus log file - The '*Consensus.log*' file records consensus at each step of Consensus Pruning. It requires disk space between 100 bytes and 1Kb per taxa and per consensus recorded. For example, recording consensus for a dataset of 200 taxa, using the metaGA heuristic for a fixed number of 5000 generations will generate a file between 100Mb and 1Gb for each replicate produced..
  - **OPDETAILS (\*\*\*)** – Operators log file - The '*OperatorsDetails.log*' file records details about the operators used. It requires disk space of 200-300 bytes per taxa per operation. For example, recording operator details for a dataset of 200 taxa, using the metaGA heuristic with 4 populations of 4 individuals each, for a fixed number of 5000 generations will generate a file between 1.7Gb and 3.4Gb for each replicate produced.
  - **OPSTATS** – Operator statistics file – The '*OperatorsStatistics.log*' file records operator statistics at the end of a search, as well as each time the operator frequencies have been updated.
  - **ANCSAQ (\*)** – Ancestral sequences log file - At the end of the heuristic, the ancestral sequence probabilities of each internal node are printed into the '*AncestralSequences.log*' file.
  - **PERF (\*)** – The '*Performances.log*' file records the amount of time (in nanoseconds) used by each operator. It requires disk space of +/- 1 Kb per iteration of the heuristic.





## 8. Appendix 2: Using the Stochastic Simulated Annealing (SSA)

Select the ‘*Simulated Annealing*’ radio button in the ‘*Heuristic*’ window to see all available parameters (Fig. 31). We implemented 14 highly-parametrized cooling schedules in MetaPIGA, including the ‘*Lundy*’ cooling schedule [26, 43]. The user can control all cooling schedule parameters: the starting temperature computation method, the maximum acceptance probability, the temperature decrease frequency, and the possibility of ‘reheating’. Changing the cooling schedule in the ‘*Heuristic*’ window will change the set of available parameters. *Note that several of these cooling schedules are quite similar to each others such that we might reduce the number of available schedules in future versions of MetaPIGA.*

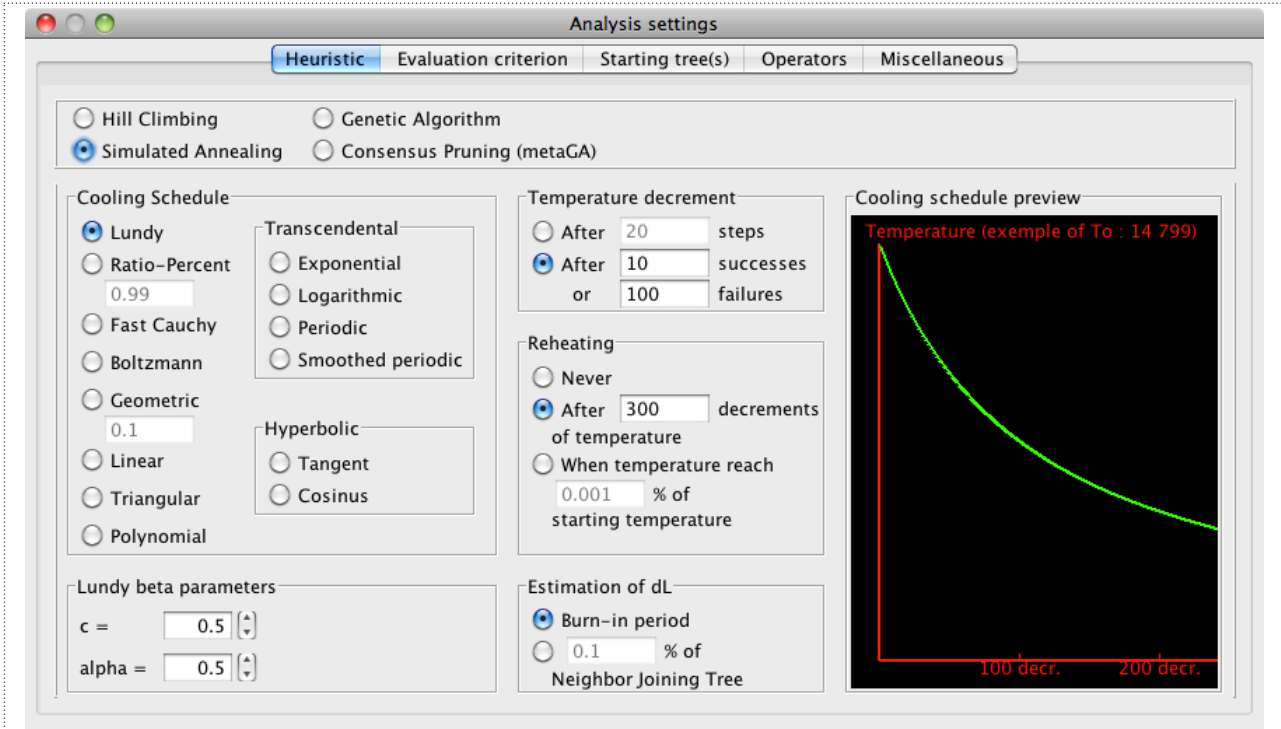


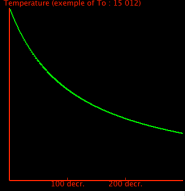
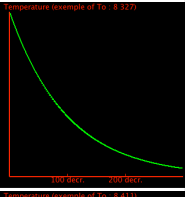
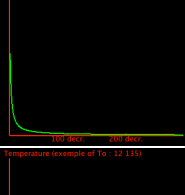
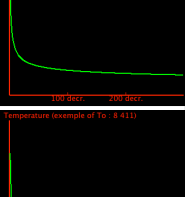
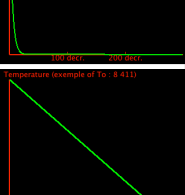
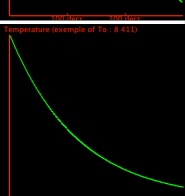
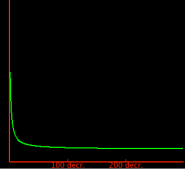
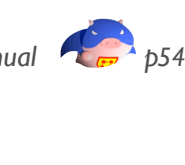
Fig. 31: The ‘*Heuristic*’ window with ‘*Simulated annealing*’ selected and the ‘*Lundy schedule*’ settings.

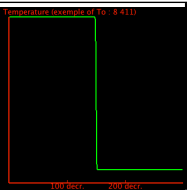
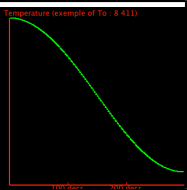
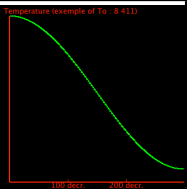
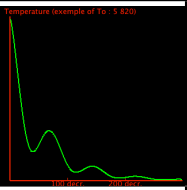
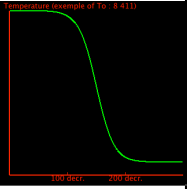
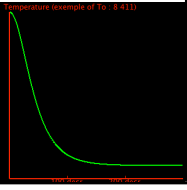
In each of the 14 available cooling schedules,  $T_i$  is the temperature after  $i$  decrements, and  $\Gamma$  is the maximum number of temperature decrements before reinitialization to  $T_0$  (the starting temperature). Except for the ‘*Lundy schedule*’,  $T_0$  (and  $T_\Gamma$  when relevant) is computed as follows:

$$T_0 = \left\lceil \frac{-\Delta L}{\ln A_0} \right\rceil \quad \text{and} \quad T_\Gamma = \left\lceil \frac{-\Delta L}{\ln A_\Gamma} \right\rceil$$

where  $\Delta L$  is the upper limit of likelihood change, whereas  $A_0$  and  $A_\Gamma$  are, respectively, the initial and final ‘*maximal acceptance parameter*’, *i.e.*, the maximal probability to accept a tree with a worse likelihood. Hence,  $A_0$  and  $A_\Gamma$  define the initial and final temperature values, and the cooling schedule defines how the temperature is decreased between these two values. The various cooling schedules (and corresponding curve equations of temperature change) are listed below, with  $A_0$  and  $A_\Gamma$  defined by the user. The cooling schedule requires defining the number of iterations (*i.e.*, the number of times operators have been used to generate a change in the tree) after which a temperature decrement is performed. The user can choose either (i) the number of iterations (steps) or (ii) the number of successes (generating better trees) or failures (not generating better trees) required before a temperature decrement is performed. As decreasing the temperature translates into rejecting more easily trees with lower likelihoods, a reheating parameter allows defining when the tem-

perature is reinitialized to  $T_0$  to facilitate crossing of valleys in likelihood space. Finally, the method for defining  $\Delta L$  (required for computing the initial and final temperatures) is also chosen by the user either as the percentage of the Likelihood of the Neighbor-Joining tree or as an estimate generated by burn-in. In the latter case, each mutation operator is applied 20 times on the starting tree and the maximum difference of likelihood observed is used as  $\Delta L$ . The table below shows the cooling schedules implemented in metaPIGA.

Cooling schedule	Corresponding curve equation	curve
<b>Lundy</b> <i>(with <math>c</math> and <math>\alpha</math> as user-defined parameters)</i>	$T_{i+1} = \frac{\Delta L}{1 + i\beta} \quad \text{with } \beta = \frac{c}{(1 - \alpha)n + \alpha \frac{-\ln NJT}{m}}$ <p><math>\beta</math> is the cooling rate (its value is <math>&lt; 1</math>) and is computed using parameters from the dataset: <math>n</math> is the number of sequences, <math>m</math> is the number of aligned columns, <math>c</math> and <math>\alpha</math> have values between 0 and 1, and <math>\ln NJT</math> is the log likelihood of the neighbour-joining tree.</p>	
<b>Ratio-Percent</b> <i>(with parameter <math>\delta</math>)</i>	$T_{i+1} = \delta T_i \quad \text{with } \delta < 1$	
<b>Fast Cauchy</b>	$T_i = \frac{T_0}{i}$	
<b>Boltzmann</b>	$T_i = \frac{T_0}{\ln i}$	
<b>Geometric</b> <i>(with parameter <math>\alpha</math>)</i>	$T_i = T_0 \alpha^i \quad \text{with } \alpha < 1$	
<b>Linear</b>	$T_i = T_0 - i \frac{(T_0 - T_\Gamma)}{\Gamma}$	
<b>Triangular</b>	$T_i = T_0 \left( \frac{T_0}{T_\Gamma} \right)^{i/\Gamma}$	
<b>Polynomial</b>	$T_i = \frac{(T_0 - T_\Gamma)(\Gamma + 1)}{\Gamma(i + 1)} + T_0 - \frac{(T_0 - T_\Gamma)(\Gamma + 1)}{\Gamma}$	

<b>Transcendental - exponential</b>	$T_i = T_\Gamma + \frac{(T_0 - T_\Gamma)}{1 + e^{3(i - \frac{\Gamma}{2})}}$	
<b>Transcendental - logarithmic</b>	$T_i = T_0 e^{-\left(\frac{i}{\Gamma}\right)^2 \ln \frac{T_0}{T_\Gamma}}$	
<b>Transcendental - periodic</b>	$T_i = \frac{(T_0 - T_\Gamma)}{2} \left(1 + \cos i \frac{\Pi}{\Gamma}\right) + T_\Gamma$	
<b>Transcendental - smoothed periodic</b>	$T_i = \frac{(T_0 - T_\Gamma)}{4} \left(2 + \cos 8i \frac{\Pi}{\Gamma}\right) e^{-\frac{i}{2\Gamma}}$	
<b>Hyperbolic - tangent</b>	$T_i = \frac{(T_0 - T_\Gamma)}{2} \left(1 - \tanh\left(\frac{10i}{\Gamma} - 5\right)\right) + T_\Gamma$	
<b>Hyperbolic - cosinus</b>	$T_i = \frac{(T_0 - T_\Gamma)}{\cosh \frac{10i}{\Gamma}} + T_\Gamma$	



## 9. Appendix 3: A simple introduction to ML phylogeny inference

### 9.1. Introduction

The Maximum Likelihood approach to phylogeny inference is based on the use of a substitution model that allows computing the likelihood of a tree, *i.e.*, the probability that its topology and branch lengths (given the model parameters, such as instantaneous substitution rates, state frequencies, gamma distribution of rates, etc) yielded the observed data. Substitution models used in the field of phylogeny inference are Markovian: the conditional probability distribution of future states depends only upon the present state, *i.e.*, the probability of change of a character from state  $i$  to state  $j$  does not depend on the history of the character before state  $i$ . We also assume that the Markov process is homogeneous (*i.e.*, the instantaneous substitution probabilities are identical everywhere in the tree) and time-reversible (the substitution rate  $i \rightarrow j$  is identical to the substitution rate  $j \rightarrow i$ ). Given time reversibility, the likelihood of a tree does not depend on where that tree is rooted. In other words, trees are unrooted and the choice of outgroup taxa (orienting the tree in time) is an assumption performed by the user. Finally, we assume that different characters (*i.e.*, different positions in the multiple alignment) evolve independently, such that the likelihood of every character can be computed separately.

### 9.2. The General-Time-Reversible (GTR) Model

The easiest way to represent a model is by using a matrix  $Q$  in which each element  $Q_{ij}$  is the instantaneous substitution rate from state  $i$  to state  $j$ . We use here the example of a 4x4 matrix for nucleotide substitutions, but the concept is the same for amino-acid substitutions or codon substitutions (but the corresponding matrices are then 20x20 and 64x64, respectively).

$$Q = \begin{pmatrix} -(\mu a \pi_C + \mu b \pi_G + \mu c \pi_T) & \mu a \pi_C & \mu b \pi_G & \mu c \pi_T \\ \mu g \pi_A & -(\mu g \pi_A + \mu d \pi_G + \mu e \pi_T) & \mu d \pi_G & \mu e \pi_T \\ \mu h \pi_A & \mu i \pi_C & -(\mu h \pi_A + \mu i \pi_C + \mu f \pi_T) & \mu f \pi_T \\ \mu j \pi_A & \mu k \pi_C & \mu l \pi_G & -(\mu j \pi_A + \mu k \pi_C + \mu l \pi_G) \end{pmatrix}$$

where  $\pi_i$  is the equilibrium frequency of state  $i$ , and  $\mu$  is the mean instantaneous substitution rate. The latter is modified with relative rate parameters  $a, b, \dots, l$  specific to each possible substitution. However, as indicated above, we use time-reversible models, such that  $a=g, b=h, c=j, d=i, e=k$ , and  $f=l$ . The diagonal elements of the matrix make the sum of each line equal to zero.

The instantaneous substitution rate matrix  $Q$  can be decomposed into a rate matrix  $R$  and an equilibrium frequency matrix  $\Pi$ :

$$Q = R \times \Pi \quad \text{Equation 2}$$

where

$$R = \begin{pmatrix} - & \mu_a & \mu_b & \mu_c \\ \mu_a & - & \mu_d & \mu_e \\ \mu_b & \mu_d & - & \mu_f \\ \mu_c & \mu_e & \mu_f & - \end{pmatrix} \quad \text{Equation 3}$$

and

$$\Pi = \begin{pmatrix} \pi_A & 0 & 0 & 0 \\ 0 & \pi_C & 0 & 0 \\ 0 & 0 & \pi_G & 0 \\ 0 & 0 & 0 & \pi_T \end{pmatrix} \quad \text{Equation 4}$$

The mean instantaneous substitution rate can be computed as follows:

$$\mu = \frac{1}{\sum_{i \neq j}^{A,C,T,G} \pi_i Q'_{ij}} \quad \text{Equation 5}$$

where

$$Q' = \begin{pmatrix} - & a\pi_C & b\pi_G & c\pi_T \\ a\pi_A & - & d\pi_G & e\pi_T \\ b\pi_A & d\pi_C & - & f\pi_T \\ c\pi_A & e\pi_C & f\pi_G & - \end{pmatrix} \quad \text{Equation 6}$$

### 9.3. Computing the likelihood of a tree

The principle for estimating the likelihood of a tree is based on computing the probability of a substitution from state  $i$  to state  $j$  (with  $i$  and  $j$  possibly identical) given the length  $v_x$  of the branch  $x$ . Given that a nucleotide in a sequence can experience multiple substitutions through time, the probability of observing a substitution between two nodes is not a linear function of the branch length  $v_x$  but takes the form:

$$p(t) = e^{-\lambda t} \quad \text{Equation 7}$$

where  $\lambda$  and  $t$  are the substitution rate and the time, respectively. Note that it is not possible to separate  $\lambda$  and  $t$  because a branch can be long due to a long time and/or a large rate. In other words, the branch length is  $\lambda t$ .

When considering the GTR model, the equation takes the form

$$p(t) = e^{Qt} \quad \text{Equation 8}$$

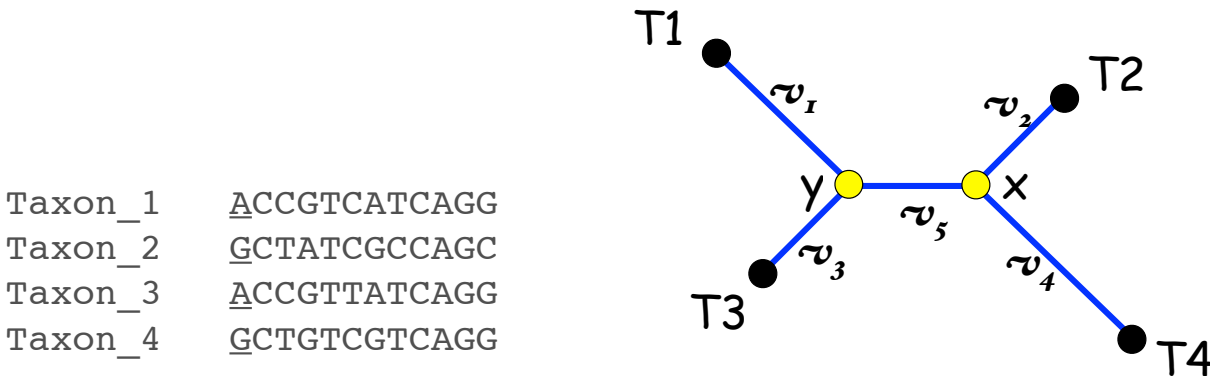


Where  $Q$  is the instantaneous substitution matrix (equation 1). The equation can be computed by using the eigenvectors and eigenvalues of the matrix.

Partitions are incorporated in the computation by multiplying  $t$  (in Equation 8) by  $\Theta_p$ , *i.e.*, the relative rate of partition  $p$ . Relative rates of partitions are optimized separately but each partition is weighted according to its size ( $S(p)$ ), and the weighted average of among-partitions rates is constrained to 1, *i.e.*,

$$\sum_p^{nPar} S(p) \cdot \Theta_p = 1 \quad \text{Equation 9}$$

Let's take a simple example. If the observed sequence data, and the tree to evaluate, are respectively:



the likelihood of that tree is the probability to generate the observed data given the substitution model. The process is performed separately for each position (each column in the alignment). Let's consider the first position (underlined in the sequence alignment above). The states at the internal nodes  $X$  and  $Y$  are unknown. Imagine that both  $X$  and  $Y$  were of state  $A$ . Given that Taxa 1 and 3 exhibit a  $A$ , and that Taxa 2 and 4 exhibit a  $G$ , the full probability of observing the first position given the tree is the Probability to:

- observe no change between  $Y(=A)$  and Taxon\_1( $=A$ ) given branch length  $v_1$
- AND observe no change between  $Y(=A)$  and Taxon\_3( $=A$ ) given branch length  $v_3$
- AND observe a change from  $X(=A)$  to Taxon\_2( $=G$ ) given branch length  $v_2$
- AND observe a change from  $X(=A)$  to Taxon\_4( $=G$ ) given branch length  $v_4$
- AND observe no change from  $X(=A)$  to  $Y(=A)$  given branch length  $v_5$

In probabilistic terms, the full probability of observing states  $A$ ,  $G$ ,  $A$ , and  $G$  for, respectively, the sequences 1, 2, 3, and 4, GIVEN that the internal nodes  $X$  and  $Y$  exhibit the state  $A$  is:

$$h(A, G, A, G \mid X=A, Y=A) = P_{AA}(v_1) \cdot P_{AA}(v_3) \cdot P_{AG}(v_2) \cdot P_{AG}(v_4) \cdot P_{AA}(v_5) \quad \text{Equation 10}$$

However, we don't know the unobserved states of the internal nodes, such that the combination considered above ( $X=Y=A$ ) is only one possibility. Hence, we have to consider each possible combination of states. In the simple tree above, there are only 2 internal nodes and 16 possibilities:

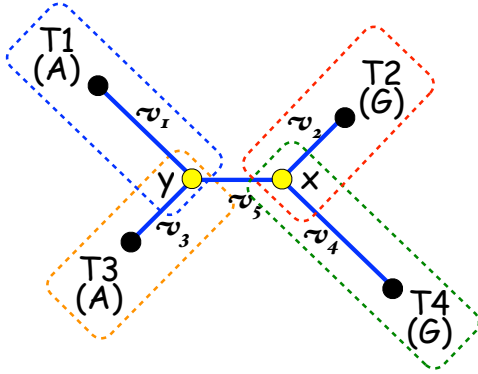
- $X=A$  and  $Y=A$  combination 1
- $X=A$  and  $Y=G$  combination 2
- $X=A$  and  $Y=C$  combination 3
- ...
- $X=T$  and  $Y=T$  combination 16

Hence, the full probability of generating the first position in the alignment above (*i.e.*, states A, G, A, and G for, respectively, the sequences 1, 2, 3, and 4) is the sum of the probabilities of combinations 1 to 16. In other words, the real (unobserved) states of the internal nodes corresponded to combination 1 or combination 2 .... or combination 16. In probabilistic terms, we therefore need to compute:

$$Prob(combination\ 1) + Prob(combination\ 2) + \dots + Prob(combination\ 16) \quad \text{Equation 11}$$

where “*Prob(combination 1)*” is equation 10.

To generalize, the likelihood of observing the first position of the alignment above given the following tree



is (equation 12):

$$h(A,G,A,G) = \sum_x g_x P_{xG}(v_4) P_{xG}(v_2) \sum_y P_{xy}(v_5) P_{yA}(v_1) P_{yA}(v_3)$$

Note the parameter  $g_x$  in equation 12, which is the equilibrium frequency of state  $x$ .

Finally, the likelihood of the tree given the full alignment is

$$L = \prod_i L_i \quad \text{Equation 13}$$

where  $L_i$  is the likelihood of position  $i$ .

To avoid the manipulation of exceedingly small values, it is much more convenient to compute the log likelihood of a tree as follows:

$$\ln L = \sum_i \ln L_i \quad \text{Equation 14}$$

Much additional information can be found in the references given in the ‘*Background*’ Section (Section 2) of this manual.

## 10. Bibliography

1. Lemmon AR, Milinkovitch MC: **The metapopulation genetic algorithm: An efficient solution for the problem of large phylogeny estimation.** *Proc Natl Acad Sci U S A* 2002, **99**:10516-10521.
2. Capella-Gutierrez S, Silla-Martinez JM, Gabaldon T: **trimAl: a tool for automated alignment trimming in large-scale phylogenetic analyses.** *Bioinformatics* 2009, **25**:1972-1973.
3. Li W-H: *Molecular evolution*. Sunderland, MA.: Sinauer; 1997.
4. Gabaldon T: **Large-scale assignment of orthology: back to phylogenetics?** *Genome Biol* 2008, **9**:235.
5. Tzika A, Helaers R, Van de Peer Y, Milinkovitch MC: **MANTIS: a phylogenetic framework for multi-species genome comparisons.** *Bioinformatics* 2008, **24**:151-157.
6. Milinkovitch MC, Helaers R, Depiereux E, Tzika AC, Gabaldon T: **2X genomes - depth does matter.** *Genome Biol* 2010, **11**:R16.
7. Thorne JL, Kishino H: **Divergence time and evolutionary rate estimation with multilocus data.** *Syst Biol* 2002, **51**:689-702.
8. Thorne JL, Kishino H, Painter IS: **Estimating the rate of evolution of the rate of molecular evolution.** *Molecular Biology and Evolution* 1998, **15**:1647-1657.
9. Cassens I, Vicario S, Waddell VG, Balchowsky H, Van Belle D, Ding W, Fan C, Mohan RS, Simoes-Lopes PC, Bastida R, et al: **Independent adaptation to riverine habitats allowed survival of ancient cetacean lineages.** *Proc Natl Acad Sci U S A* 2000, **97**:11343-11347.
10. Chang BS, Jonsson K, Kazmi MA, Donoghue MJ, Sakmar TP: **Recreating a functional ancestral archosaur visual pigment.** *Mol Biol Evol* 2002, **19**:1483-1489.
11. Chang BS, Kazmi MA, Sakmar TP: **Synthetic gene technology: applications to ancestral gene reconstruction and structure-function studies of receptors.** *Methods Enzymol* 2002, **343**:274-294.
12. Chang BS, Ugalde JA, Matz MV: **Applications of ancestral protein reconstruction in understanding protein function: GFP-like proteins.** *Methods Enzymol* 2005, **395**:652-670.
13. Blanchette M, Green ED, Miller W, Haussler D: **Reconstructing large regions of an ancestral mammalian genome in silico.** *Genome Res* 2004, **14**:2412-2423.
14. Williams PD, Pollock DD, Blackburne BP, Goldstein RA: **Assessing the accuracy of ancestral protein reconstruction methods.** *PLoS Comput Biol* 2006, **2**:e69.
15. Zhang J, Nielsen R, Yang Z: **Evaluation of an improved branch-site likelihood method for detecting positive selection at the molecular level.** *Mol Biol Evol* 2005, **22**:2472-2479.
16. Meegaskumbura M, Bossuyt F, Pethiyagoda R, Manamendra-Arachchi K, Bahir M, Milinkovitch MC, Schneider CJ: **Sri Lanka: an amphibian hot spot.** *Science* 2002, **298**:379.
17. Springer MS, Stanhope MJ, Madsen O, de Jong WW: **Molecules consolidate the placental mammal tree.** *Trends Ecol Evol* 2004, **19**:430-438.
18. Bossuyt F, Brown RM, Hillis DM, Cannatella DC, Milinkovitch MC: **Phylogeny and biogeography of a cosmopolitan frog radiation: Late cretaceous diversification resulted in continent-scale endemism in the family ranidae.** *Syst Biol* 2006, **55**:579-594.
19. Graham RL, Foulds LR: **Unlikelihood that Minimal Phylogenies for a Realistic Biological Study Can Be Constructed in Reasonable Computational Time.** *Math Bioscience* 1982, **60**:133-142.
20. Chor B, Tuller T: **Maximum likelihood of evolutionary trees: hardness and approximation.** *Bioinformatics* 2005, **21 Suppl 1**:i97-106.
21. Felsenstein J: *Inferring Phylogenies*. Sunderland: Sinauer Associates Inc.; 2004.
22. Felsenstein J: **Evolutionary trees from DNA sequences: a maximum likelihood approach.** *Journal of Molecular Evolution* 1981, **17**:368-376.
23. Swofford DL, Waddell PJ, Huelsenbeck JP, Foster PG, Lewis PO, Rogers JS: **Bias in phylogenetic estimation and its relevance to the choice between parsimony and likelihood methods.** *Syst Biol* 2001, **50**:525-539.
24. Huelsenbeck JP, Larget B, Miller RE, Ronquist F: **Potential applications and pitfalls of Bayesian inference of phylogeny.** *Syst Biol* 2002, **51**:673-688.
25. Holder M, Lewis PO: **Phylogeny estimation: traditional and Bayesian approaches.** *Nat Rev Genet* 2003, **4**:275-284.
26. Salter LA, Pearl DK: **Stochastic search strategy for estimation of maximum likelihood phylogenetic trees.** *Syst Biol* 2001, **50**:7-17.
27. Matsuda H: **Protein phylogenetic inference using maximum likelihood with a genetic algorithm.** In *Pacific symposium on biocomputing '96; London*. Edited by Hunter L, Klein TE. World Scientific; 1996: 512-523.
28. Katoh K, Kuma K, Miyata T: **Genetic algorithm-based maximum-likelihood analysis for molecular phylogeny.** *J Mol Evol* 2001, **53**:477-484.
29. Lewis PO: **A genetic algorithm for maximum-likelihood phylogeny inference using nucleotide sequence data.** *Mol Biol Evol* 1998, **15**:277-283.
30. Zwickl DJ: **Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion.** The University of Texas, 2006.
31. Ronquist F, Huelsenbeck JP: **MrBayes 3: Bayesian phylogenetic inference under mixed models.** *Bioinformatics* 2003, **19**:1572-1574.
32. Stamatakis A: **RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models.** *Bioinformatics* 2006, **22**:2688-2690.
33. Suchard MA, Rambaut A: **Many-core algorithms for statistical phylogenetics.** *Bioinformatics* 2009, **25**:1370-1376.
34. Tavaré S: **Some Probabilistic and Statistical Problems in the Analysis of DNA Sequences.** *American Mathematical Society: Lectures on Mathematics in the Life Sciences* 1986, **17**:57-86.
35. Yang Z: **Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods.** *J Mol Evol* 1994, **39**:306-314.

36. Yang Z: **Among-site rate variation and its impact on phylogenetic analyses.** *Trends in Ecology & Evolution* 1996, **11**:367-372.
37. Gu X, Fu YX, Li WH: **Maximum likelihood estimation of the heterogeneity of substitution rate among nucleotide sites.** *Mol biol evol* 1995, **12**:546-557.
38. Guindon S, Gascuel O: **A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood.** *Syst Biol* 2003, **52**:696-704.
39. Stamatakis A, Ludwig T, Meier H: **RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees.** *Bioinformatics* 2005, **21**:456-463.
40. Maddison DR, Swofford DL, Maddison WP: **NEXUS: an extensible file format for systematic information.** *Syst Biol* 1997, **46**:590-621.
41. Posada D, Crandall KA: **Selecting the best-fit model of nucleotide substitution.** *Syst Biol* 2001, **50**:580-601.
42. Kirkpatrick S, Gelatt CD, Jr., Vecchi MP: **Optimization by Simulated Annealing.** *Science* 1983, **220**:671-680.
43. Lundy M: **Applications of the Annealing Algorithm to Combinatorial Problems in Statistics.** *Biometrika* 1985, **72**:191-198.
44. Holland J: *Adaptation in Natural and Artificial Systems.* Ann Arbor: University of Michigan Press; 1975.
45. Goldman N, Yang Z: **A codon-based model of nucleotide substitution for protein-coding DNA sequences.** *Mol Biol Evol* 1994, **11**:725-736.
46. Kosiol C, Holmes I, Goldman N: **An empirical codon model for protein sequence evolution.** *Mol biol evol* 2007, **24**:1464-1479.
47. Saitou N, Nei M: **The neighbor-joining method: a new method for reconstructing phylogenetic trees.** *Molecular Biology and Evolution* 1987, **4**:406-425.
48. Criscuolo A, Michel CJ: **Phylogenetic inference with weighted codon evolutionary distances.** *J Mol Evol* 2009, **68**:377-392.
49. Huelsenbeck JP, Bollback JP: **Empirical and hierarchical Bayesian estimation of ancestral states.** *Syst Biol* 2001, **50**:351-366.